**Today**

- Quiz 12
- More ASP

**Assignments**

- HW 11 out
- Extra credit 2 out (due by end of semester)
- Proj Presentations next Thurs
REVIEW: Informally, Answer Sets ...

1. Satisfy all rules of the program ...
   - if the answer includes the body it also includes the head
   - if the answer doesn’t include the body, the rule is trivially satisfied

2. Do not contain contradictions (e.g., $p$ and $\neg p$)

3. Are minimal
   - the *Rationality Principle*: “Believe nothing you are not forced to believe"
   - always forced to believe asserted facts
   - each answer set is as small as it can be
Q: What is the “answer” (answer set) of the following?

In propositional logic:

\[
  q \leftarrow p \\
  r \leftarrow q
\]

In LP syntax:

\[
  q \iff p \\
  r \iff q
\]

A: \{

- Recall the Rationality Principle
- ... this answer supports the criteria for being minimal

Some other possible models ... (but not answer sets)

- \{p, q, r\} ... but \{\} \subseteq \{p, q, r\}
- \{q, r\} ... but \{\} \subseteq \{q, r\}
- What about \{p, q\}? ... no since \( r \leftarrow q \)

Q: What is the “answer” (answer set) of the following?

In propositional logic:

\[
  q \leftarrow p \\
  \neg p \leftarrow q \\
  p
\]

In LP syntax:

\[
  q \iff p \\
  \neg p \iff q \\
  p
\]

This is not satisfiable ... there is no answer set

- \{p, q, \neg p\} contains a contradiction \((p \land \neg p)\)
Classical Negation (¬)

Q: What is the “answer” (answer set) of the following?

\[ \begin{align*}
  v & \leftarrow \neg s \land \neg r \\
  \neg s & \leftarrow \neg u \\
  \neg r & \leftarrow \neg u \\
  \neg u &
\end{align*} \]

In propositional logic:

\[ \begin{align*}
  v & :> > s = > r. \\
  \neg s & :> > u. \\
  \neg r & :> > u. \\
  \neg u &
\end{align*} \]

In LP syntax:

Note: \( s = \text{sunny}, \ r = \text{raining}, \ v = \text{overcast}, \ u = \text{umbrella} \)

A: \( \{ \neg u, \neg r, \neg s, v \} \)
(Epistemic) Disjunction (∨)

Q: What is the “answer” (answer set) of the following?

In propositional logic:  
\[ p \lor q \]

In LP syntax:  
\[ p \mid q. \]

This example has 2 answer sets: \{p\}, \{q\}

• Note that \{p, q\} is not a minimal answer

• Note that \( p \mid q \) and \( p \lor q \) are different (e.g., \( p \mid -p \) is not a tautology)

• To obtain all answer sets, run: clingo file.lp 0

Q: What is the “answer” (answer set) of the following?

In propositional logic:  
\[ r \lor s \leftarrow p \]
\[ r \lor s \leftarrow q \]
\[ p \]
\[ q \]

In LP syntax:  
\[ r \mid s :- p. \]
\[ r \mid s :- q. \]
\[ p. \]
\[ q. \]

This example has 2 answer sets: \{p, q, r\}, \{p, q, s\}

Allows us to represent “incomplete” knowledge ...

• if we know it’s not overcast then it’s sunny or raining (but don’t know which)

\[ \text{sunny} \mid \text{raining} :- \neg\text{-overcast}. \]
Default Negation (not)

- What can’t be proved I believe to be false "negation as failure"

Q: What is the “answer” (answer set) of the following?

\[
\begin{align*}
u & : \neg v. & \text{... need umbrella if can’t prove overcast} \\
u & : r. & \text{... need umbrella if raining} \\
u & : s. & \text{... need umbrella if sunny} \\
v & : \neg r, \neg s. & \text{... overcast if no rain and no sun}
\end{align*}
\]

A: \{u\} \text{ ... Why?}

- \{\} doesn’t satisfy rules since it makes \(\neg v\) true
- \{u\} satisfies all of the rules
- \{r, u\} and \{s, u\} aren’t minimal
- nothing else follows rationality principle
Q: What is the “answer” (answer set) of the following?

\[
\begin{align*}
p & : \neg q. \\
q & : \neg p. \\
r & : p. \\
s & : p.
\end{align*}
\]

There are two: \{p, r, s\} and \{q\} ... Why?

**Exercise:** Come up with an example for these rules ... (replace symbols)

```prolog
% facts about Garfield
full :- not hungry.
hungry :- not full.
happy :- full.
asleep :- full.
```

Possible outcomes are that Garfield is either hungry or else full, happy, and asleep.
Some more basic terminology

- A proposition symbol (e.g., p, q, r) is called an **atom** (for “atomic statement”)
- A **literal** is an atom or its negation
- E.g., p is an atom, whereas both p and ¬p are literals

Dealing with default negation

Assume Π is a program and S is a set of (ground) literals

Π^S is obtained from Π by:

1. removing all rules containing not l for l ∈ S
2. removing all other premises containing not from rules

S is an answer set of Π if S is an answer set of Π^S.

Example: Assume S = {p, t}

<table>
<thead>
<tr>
<th>Original program (Π)</th>
<th>Rewritten program (Π^S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>q :- not p.</td>
<td>% removed since p in S</td>
</tr>
<tr>
<td>t :- p, not r.</td>
<td>t :- p.</td>
</tr>
</tbody>
</table>

{p, t} is an answer set of Π since it is an answer set of Π^S
Exercise: What is the answer set for the following program?

\[
\begin{align*}
p &: \neg q. \\
q &: \neg p. \\
p &: q. \\
q &: p.
\end{align*}
\]

Hint: The only possibilities are {}, {p}, {q}, {p, q}, or unsatisfiable

- $S = \{\}$ doesn’t satisfy first two rules
- $S = \{p\}$ is not an answer set of $\Pi^S$:
  
  \[
  \begin{align*}
p. & \quad \text{% remove } \neg q \text{ (since } q \text{ not in } S) \\
q &: p. & \text{% 2nd rule removed (since } p \text{ in } S) \\
p &: q. & \text{% trivially satisfied by } \{p\} \\
q &: p. & \text{% not satisfied by } \{p\}
\end{align*}
  \]

- $S = \{q\}$ is also not an answer set of $\Pi^S$
- $S = \{p, q\}$ is also not an answer set of $\Pi^S$:

  \[
  \begin{align*}
p &: q. & \text{% 1st rule removed (since } q \text{ in } S) \\
q &: p. & \text{% 2nd rule removed (since } p \text{ in } S) \\
p &: q. & \text{% satisfied by } \{p, q\} \\
q &: p. & \text{% satisfied by } \{p, q\}
\end{align*}
  \]

However, the answer set of $\Pi^S$ is \{\}  \quad \ldots \quad \text{so, } \{\} \neq S

This program is unsatisfiable (doesn’t have an answer set)
From Propositional to Predicate Logic

In predicate logic, an **atom** is of the form:

\[ p(t_1, t_2, \ldots, t_n) \]

More terminology:
- \( p \) is a **predicate**
- the **arity** of \( p \) above is \( n \) ... written \( p/n \)
- each \( t_i \) is a **term** ... a constant, variable, or “function”
- a proposition is still an atom (with arity 0)

Examples:
- \( \text{edge}(a,b) \) % a and b are constants
- \( \text{child_of}(\text{luke}, \text{padme}) \) % child_of has arity 2
- \( \text{child_of}(\text{luke}, X) \) % \( X \) is a variable
- \( \text{employee}(\text{alice}, \text{google}, \text{engineer}) \) % employee/3
- \( \text{enrolled_in}(\text{bob}, \text{ssn}(\text{bob}), 326) \) % \( \text{ssn} \) is a function, rest constants
- \( \text{list}((\text{cons}(1, \text{cons}(2, \text{nil})))) \) % nested functions

A term is **ground** if it doesn’t contain variables

An atom is **ground** if all of its terms are ground
Q: What is the “answer” (answer set) of the following?

\[
\begin{align*}
\text{grand\_child\_of}(a, c) & :\text{ child\_of}(a, b), \text{ child\_of}(b, c). \\
\text{child\_of}(a, b). \\
\text{child\_of}(b, c).
\end{align*}
\]

A: \{\text{child\_of}(a, b), \text{child\_of}(b, c), \text{grand\_child\_of}(a, c)\}

Q: What is the “answer” (answer set) of the following?

\[
\begin{align*}
\text{make}(\text{bob, honda}) & \mid \text{make}(\text{bob, toyota}). \\
\text{color}(\text{bob, red}) & \mid \text{color}(\text{bob, green}). \\
\text{color}(\text{bob, red}) & :\text{ make}(\text{bob, toyota}). \ % \text{no green toyota's}
\end{align*}
\]

A: There are three ...

- \{\text{make}(\text{bob, honda}), \text{color}(\text{bob, red})\}
- \{\text{make}(\text{bob, honda}), \text{color}(\text{bob, green})\}
- \{\text{make}(\text{bob, toyota}), \text{color}(\text{bob, red})\}
Dealing with Variables

1. Only rules (with a nonempty body) can contain variables
   - there are additional “safety” constraints
   - e.g., \( q(X) :\not p(X) \) if \( X \) not used in a positive body literal
   - and \( q(X) :\not p(X) \) if \( X \) not used in the rule body

2. Answer sets are computed over a “ground” program (no variables) ...
   - Each rule with variables is replaced by its “ground instantitations”
   - A ground instantiation has each variable replaced by a ground term
   - Thus, a rule with variables will be replaced by many ground instances

Example:

<table>
<thead>
<tr>
<th>Original program</th>
<th>Ground program</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p(a) ).</td>
<td>( p(a) ).</td>
</tr>
<tr>
<td>( p(b) ).</td>
<td>( p(b) ).</td>
</tr>
<tr>
<td>( q(X) :\not p(X) ).</td>
<td>( q(a) :\not p(a) ).</td>
</tr>
<tr>
<td></td>
<td>( q(b) :\not p(b) ).</td>
</tr>
</tbody>
</table>

ASP systems usually involve two steps: grounding followed by solving
- The grounding procedure heavily influences performance
- Various optimizations developed to reduce set of ground rule instantiations
Integrity Constraints

Constraints act as filters on entire answer sets not parts of answer sets.

A constraint takes the form:

$$\text{:- } l_1, l_2, \ldots, l_n$$

The set $S$ satisfies the constraint if at least one $l_i \notin S$.

- in other words, the body of the constraint must not be satisfied.
- if it is satisfied, $S$ is not an answer set.

Q: What is the “answer” (answer set) of the following?

$$\text{make(bob, honda) | make(bob, toyota).}$$
$$\text{color(bob, red) | color(bob, green).}$$
$$\text{:- color(X, green), make(X, toyota).} \quad \% \text{no green toyota's}$$

A: the same three as before.

Constraints are not technically needed ...

$$\text{fail :- } l_1, l_2, \ldots, l_n.$$ 
$$\text{-fail.}$$

If the rule body is satisfied, fail must be in $S$, resulting in a contradiction.
A choice rule is extra syntax for disjunctive rules (make "guesses")

Example:

\[
\text{color(r). color(g). color(b). color(y). ... etc ...}
\]

\[1 \{\text{has\_color(X,C) : color(C)}\} 1 :- \text{node(X)}.
\]

Result is an answer set for every possible assignment of a color to a node

- \{\text{has\_color(1,r), has\_color(2,r), ...}\}
- \{\text{has\_color(1,r), has\_color(2,g), ...}\}
- and so on

A choice rule takes the form:

\[
n1 \{p(X) : q(X)\} n2 :- \text{body.}
\]

where \(n1, n2\) optional

which makes a set \(C\) of "guesses" of \(p(t)\) atoms such that

- \(n1 \leq |C| \leq n2\) without \(n1\) and \(n2\), \(0 \leq |S|\)
- for each body, if \(q(t) \in S\) then \(p(t)\) is one of the possible choices

Q: What are the answer sets?

\[
0 \{ \text{has\_color(X,C)} \} 1 :- \text{node(X), color(C)}.
\]

A: for each \(X-C\) combination, \text{has\_color(X,C)} is either asserted or not asserted
clingo-supported aggregate functions

- \#count, \#sum, \#min, and \#max

The basic syntax of an aggregate atom
\[ v_1 \rel_1 f \{ t_1:L_1; t_2:L_2; \ldots; t_n:L_n \} \rel_2 v_2 \]

where:
- \( f \) is one of the aggregate functions
- \( v_1 \) and \( v_2 \) are integer values (or variables)
- \( \rel_1 \) and \( \rel_2 \) are comparators (\(<\), \(\geq\), \(\leq\), \(\neq\))
- \( v_1 \rel_1 \) and \( \rel_2 v_2 \) are optional
- \( t_i \) is a term “tuple” and \( L_i \) is a literal “tuple”
- aggregate function works over the set of term tuples

Examples

The sum of the unique term “weights” \{1,1,2,2\}

\[
\#\text{sum} \{ 1:p; 1:q; 2:p; 2:q \} \quad \text{evaluates to 3 if } p,q \text{ are believed}
\]

Sums of the T values for unique T,F pairs (from xkcd example)

\[
1505 \neq \#\text{sum} \{ \text{T,F : total_order(F,N,T)} \}
\]

Number of unique X,Y pairs (to count the unique graph edges)

\[
\text{edges}(N) :- N = \#\text{count} \{ \text{X,Y : edge}(X,Y) \}.
\]

Number of other characters a particular character X speaks with

\[
N = \#\text{count} \{ Y : \text{speaks_with}(X,Y) \}, \text{character}(X,\_)
\]
Intervals and Pooling

- Intervals: `size(1..3)` is replaced by `size(1). size(2). size(3).
- Pooling: `color(red ; blue)` is replaced by `color(red). color(blue).

Recursion: Computing Transitive Closures

```
descendent(X,Y) :- child_of(X,Y).
descendent(X,Y) :- child_of(X,Z), descendent(Z,Y).
```

Note that order of literals and rules doesn't matter!