Answer Set Programming

CMPT 411/721

(based on slides by Torsten Schaub)
Introduction:
Model-Based Problem Solving
Goal: Declarative problem solving

In declarative problem solving:

- Instead of asking: "How can the problem be solved?"
- Ask: "How can the problem be described?"

Then use a domain-independent solver to compute a solution
Goal: Declarative problem solving

In declarative problem solving:

- Instead of asking: "How can the problem be solved?"
- Ask: "How can the problem be described?"

Then use a domain-independent solver to compute a solution

General KR Methodology:
Answer set programming (ASP)

- Has its roots in
  - Knowledge representation and reasoning
    - In particular nonmonotonic reasoning
  - Deductive databases (particularly Datalog)
  - Constraint solving (in particular, SAT solving)
  - Logic programming (with negation)
Answer set programming (ASP)

- Has its roots in
  - Knowledge representation and reasoning
    - In particular nonmonotonic reasoning
  - Deductive databases (particularly Datalog)
  - Constraint solving (in particular, SAT solving)
  - Logic programming (with negation)

- Allows for solving all search problems within NP (and $NP^{NP}$) (over finite domains).
Answer set programming (ASP)

- Has its roots in
  - Knowledge representation and reasoning
    - In particular nonmonotonic reasoning
  - Deductive databases (particularly Datalog)
  - Constraint solving (in particular, SAT solving)
  - Logic programming (with negation)

- Allows for solving all search problems within NP (and $NP^{NP}$) (over finite domains).

- Allows for using powerful off-the-shelf systems
  (nowadays capable of dealing with millions of variables)
**Example: 3–colourability of graphs**

<table>
<thead>
<tr>
<th>C(I)</th>
<th>vertex(1) ← edge(1,2) ← vertex(2) ← edge(2,3) ← vertex(3) ← edge(3,1) ←</th>
</tr>
</thead>
<tbody>
<tr>
<td>C(P)</td>
<td>coloured(V,r) ← not coloured(V,b), not coloured(V,g), vertex(V)</td>
</tr>
<tr>
<td></td>
<td>coloured(V,b) ← not coloured(V,r), not coloured(V,g), vertex(V)</td>
</tr>
<tr>
<td></td>
<td>coloured(V,g) ← not coloured(V,r), not coloured(V,b), vertex(V)</td>
</tr>
<tr>
<td></td>
<td>← edge(V,U), coloured(V,C), coloured(U,C), colour(C)</td>
</tr>
</tbody>
</table>

Answer set  { coloured(1,r), coloured(2,b), coloured(3,g), … }

Goal: Find a *minimal* set of literals that *satisfies* the rules.

Such a set of literals is called an *answer set*
Model-Based Problem Solving

Compare:

I Inference-based approach

1. Provide a specification of the problem.
2. A solution is given by a derivation of an appropriate query.
   - E.g. resolution in logic, top-down rule-based reasoning, Prolog

II Model-based approach

1. Provide a specification of the problem.
2. A solution is given by a model of the specification.
   - E.g. ASP, also SAT

Key Idea: Rules represent constraints on the problem.
Model-Based Problem Solving

Compare:

I Inference-based approach

1. Provide a specification of the problem.
2. A solution is given by a derivation of an appropriate query.
   - E.g. resolution in logic, top-down rule-based reasoning, Prolog

II Model-based approach

1. Provide a specification of the problem.
2. A solution is given by a model of the specification.
   - E.g. ASP, also SAT
Model-Based Problem Solving

Compare:

I Inference-based approach

1. Provide a specification of the problem.
2. A solution is given by a derivation of an appropriate query.
   • E.g. resolution in logic, top-down rule-based reasoning, Prolog

II Model-based approach

1. Provide a specification of the problem.
2. A solution is given by a model of the specification.
   • E.g. ASP, also SAT

Key Idea: Rules represent constraints on the problem.
Applications of ASP

- Combinatorial search problems:
  - auctions, bio-informatics, computer-aided verification, configuration, constraint satisfaction, diagnosis, information integration, planning and scheduling, security analysis, semantic web, wire-routing, zoology and linguistics, ...

- ASP has also been used as a target language into which a high level language can be compiled.
  - E.g.: Action language $\Rightarrow$ ASP
Introduction to ASP
ASP: Idea

- A (normal) rule, $r$, is of the form
  
  $A_0 \leftarrow A_1, \ldots, A_m, \textit{not } A_{m+1}, \ldots, \textit{not } A_n,$

- \textit{not} can be read as negation as failure.
- Variables are treated as standing for all possible instances.

\[
\{a \leftarrow b, \textit{not } c, b\}\text{ has answer set } \{a, b\}.
\]

\[
\{a \leftarrow \textit{not } b, b \leftarrow \textit{not } a\}\text{ has answer sets } \{a\} \text{ and } \{b\}.
\]
ASP: Idea

- A (normal) rule, \( r \), is of the form
  \[
  A_0 \leftarrow A_1, \ldots, A_m, \neg A_{m+1}, \ldots, \neg A_n,
  \]
- \( \neg \) can be read as negation as failure.
- Variables are treated as standing for all possible instances.
- Want to determine answer sets of a set of rules, or program.
- An answer set is a minimal set of atoms satisfying the rules.
  - I.e. for rule \( r \) above, if \( X \) is an answer set, then if \( A_1, \ldots, A_m \) are in \( X \) and no \( A_{m+1}, \ldots, A_n \) is in \( X \) then \( A_0 \) is in \( X \).
ASP: Idea

• A (normal) rule, \( r \), is of the form

\[
A_0 \leftarrow A_1, \ldots, A_m, \text{not } A_{m+1}, \ldots, \text{not } A_n,
\]

• \text{not} can be read as negation as failure.
• Variables are treated as standing for all possible instances.

• Want to determine answer sets of a set of rules, or program.

• An answer set is a minimal set of atoms satisfying the rules.
  • I.e. for rule \( r \) above, if \( X \) is an answer set, then if \( A_1, \ldots, A_m \) are in \( X \) and no \( A_{m+1}, \ldots, A_n \) is in \( X \) then \( A_0 \) is in \( X \).
• E.g. \{a \leftarrow b, \text{not } c., \ b.\} \text{ has answer set } \{a, b\}.
A (normal) rule, $r$, is of the form

$$A_0 \leftarrow A_1, \ldots, A_m, \text{ not } A_{m+1}, \ldots, \text{ not } A_n,$$

- *not* can be read as negation as failure.
- Variables are treated as standing for all possible instances.

Want to determine answer sets of a set of rules, or program.

An answer set is a minimal set of atoms satisfying the rules.

- I.e. for rule $r$ above, if $X$ is an answer set, then if $A_1, \ldots, A_m$ are in $X$ and no $A_{m+1}, \ldots, A_n$ is in $X$ then $A_0$ is in $X$.

E.g. $\{a \leftarrow b, \text{ not } c., \ b.\}$ has answer set $\{a, b\}$.

$\{a \leftarrow \text{ not } b., \ b \leftarrow \text{ not } a.\}$ has answer sets $\{a\}$ and $\{b\}$. 
ASP: Atoms and Terms

Atoms

• An atom is the elementary construct for representing knowledge
• An atom in general represents a relation between objects
• Examples: \textit{answer}(42), \textit{coloured}(1, \textit{red}), \textit{hot}
• An atom can be either true or false

Terms

• Terms are the subatomic components of atoms
• Terms represent objects
  - Examples: 42, \textit{red}, \textit{joe}
• Variables are also terms, but are removed from a program by \textit{grounding} the program
Normal logic programs

- A (normal) logic program is a finite set of rules.
Normal logic programs

- A (normal) logic program is a finite set of rules.
- A (normal) rule, \( r \), is of the form
  
  \[
  A_0 \leftarrow A_1, \ldots, A_m, \text{not } A_{m+1}, \ldots, \text{not } A_n,
  \]

  where \( n, m \geq 0 \), and each \( A_i \) is an atom.
Normal logic programs

- A (normal) logic program is a finite set of rules.
- A (normal) rule, \( r \), is of the form
  \[ A_0 \leftarrow A_1, \ldots, A_m, \text{not } A_{m+1}, \ldots, \text{not } A_n, \]
  where \( n, m \geq 0 \), and each \( A_i \) is an atom.
- Notation
  \[
  \begin{align*}
  \text{head}(r) &= A_0 \\
  \text{body}(r) &= \{A_1, \ldots, A_m, \text{not } A_{m+1}, \ldots, \text{not } A_n\} \\
  \text{body}^+(r) &= \{A_1, \ldots, A_m\} \\
  \text{body}^-(r) &= \{A_{m+1}, \ldots, A_n\}
  \end{align*}
  \]
Normal logic programs

- A (normal) logic program is a finite set of rules.
- A (normal) rule, \( r \), is of the form
  \[ A_0 \leftarrow A_1, \ldots, A_m, \text{not} \ A_{m+1}, \ldots, \text{not} \ A_n, \]
  where \( n, m \geq 0 \), and each \( A_i \) is an atom.
- Notation
  \[
  \begin{align*}
  \text{head}(r) & = A_0 \\
  \text{body}(r) & = \{A_1, \ldots, A_m, \text{not} \ A_{m+1}, \ldots, \text{not} \ A_n\} \\
  \text{body}^+(r) & = \{A_1, \ldots, A_m\} \\
  \text{body}^-(r) & = \{A_{m+1}, \ldots, A_n\}
  \end{align*}
  \]
- A program is called positive if \( \text{body}^-(r) = \emptyset \) for all its rules.

\( \{ \} = \text{set of Horn clauses} \)
Examples of Rules

Examples

• a :- b, not c.
• a :- not c, b.
• a.
• a :- b.
• a :- not c.
• ugrad(joe) :- student(joe), not grad(joe).
• ugrad(X) :- student(X), not grad(X).
## Notational Conventions

The following notation is used interchangeably in order to stress a particular view:

<table>
<thead>
<tr>
<th></th>
<th>if</th>
<th>and</th>
<th>or</th>
<th>negation as failure</th>
<th>classical negation</th>
</tr>
</thead>
<tbody>
<tr>
<td>logic program</td>
<td>←</td>
<td>,</td>
<td>;</td>
<td>not/~</td>
<td>¬</td>
</tr>
<tr>
<td>formula</td>
<td>→</td>
<td>∧</td>
<td>∨</td>
<td></td>
<td>¬</td>
</tr>
<tr>
<td>source code</td>
<td>:-</td>
<td>,</td>
<td></td>
<td>not</td>
<td></td>
</tr>
</tbody>
</table>
Answer Set: Intuitions

- An *answer set* for a program $P$ is a *minimal* set of atoms $X$ such that, for every rule:

$$A_0 \leftarrow A_1, \ldots, A_m, \text{not } A_{m+1}, \ldots, \text{not } A_n,$$

if

$$\{A_1, \ldots, A_m\} \subseteq X$$

and

$$\{A_{m+1}, \ldots, A_n\} \cap X = \emptyset$$

then

$$A_0 \in X.$$

- This is a *nonconstructive* specification.
- Think of rules as specifying *constraints* on an answer set.
Classical Logic

- A model in classical logic can be written as a set of atoms $X$ where
  - atoms in $X$ are \textit{true} and
  - atoms not in $X$ are \textit{false}.
- The formula $\neg b \rightarrow a$ has models $\{a\}$, $\{b\}$, and $\{a, b\}$,
Answer sets and models

Classical Logic

- A model in classical logic can be written as a set of atoms $X$ where
  - atoms in $X$ are \textit{true} and
  - atoms not in $X$ are \textit{false}.
- The formula $\neg b \rightarrow a$ has models $\{a\}$, $\{b\}$, and $\{a, b\}$.

ASP

- An answer set $X$ can be regarded as a model where
  - atoms in $X$ are \textit{true} and
  - atoms not in $X$ are \textit{false}.
- Program $\{a \leftarrow \text{not } b\}$ has answer set $\{a\}$.

💡 The negation-as-failure operator \textit{not} makes a difference!
Answer sets: Basic idea

Consider the set of formulas: \( \{ q, (q \land \neg r) \rightarrow p \} \)
Answer sets: Basic idea

Consider the set of formulas: \( \{q, (q \land \neg r) \rightarrow p\} \)

This set has three (classical) models: \( \{p, q\}, \{q, r\}, \{p, q, r\} \).
Answer sets: Basic idea

Consider the set of formulas: \( \{ q, (q \land \neg r) \rightarrow p \} \)

This set has three (classical) models: \( \{ p, q \}, \{ q, r \}, \{ p, q, r \} \).

The corresponding logic program is:

\[
q \leftarrow \\
p \leftarrow q, \ not \ r
\]
Consider the set of formulas: \( \{ q, (q \land \neg r) \rightarrow p \} \)

This set has three (classical) models: \( \{ p, q \}, \{ q, r \}, \{ p, q, r \} \).

The corresponding logic program is:

\[
q \leftarrow \\
p \leftarrow q, \text{not } r
\]

This logic program has one answer set: \( \{ p, q \} \)
Answer sets: Basic idea

Consider the set of formulas: \( \{ q, (q \land \neg r) \rightarrow p \} \)

This set has three (classical) models: \( \{ p, q \} \), \( \{ q, r \} \), \( \{ p, q, r \} \).

The corresponding logic program is:

\[
\begin{align*}
q & \leftarrow \\
p & \leftarrow q, \text{not } r
\end{align*}
\]

This logic program has one answer set: \( \{ p, q \} \)

Roughly, a set of atoms \( X \) is an answer set of a logic program \( \Pi \) if

- \( X \) is a (classical) model of \( \Pi \) and
- all atoms in \( X \) are justified by some rule in \( \Pi \)

\( \star \) Each atom in \( X \) is a fact or is the head of a satisfied rule.
Answer Set: Formal Definition

Positive programs

- A set of atoms $X$ is **closed under** a positive program $\Pi$ iff for any $r \in \Pi$: if $\text{body}^+(r) \subseteq X$ then $\text{head}(r) \in X$.
  - $X$ corresponds to a model of $\Pi$ (seen as a formula).
Answer Set: Formal Definition

Positive programs

- A set of atoms $X$ is **closed under** a positive program $\Pi$ iff for any $r \in \Pi$: if $\text{body}^+(r) \subseteq X$ then $\text{head}(r) \in X$.
  
  $\implies X$ corresponds to a model of $\Pi$ (seen as a formula).

- The **smallest** set of atoms which is closed under a positive program $\Pi$ is denoted by $Cn(\Pi)$.
  
  - $Cn(\Pi)$ corresponds to the $\subseteq$-smallest model of $\Pi$
  - This is the set of consequences obtained by forward chaining.
Answer Set: Formal Definition

Positive programs

- A set of atoms $X$ is **closed under** a positive program $\Pi$ iff for any $r \in \Pi$: if $\text{body}^+(r) \subseteq X$ then $\text{head}(r) \in X$.
  
  $X$ corresponds to a model of $\Pi$ (seen as a formula).

- The **smallest** set of atoms which is closed under a positive program $\Pi$ is denoted by $Cn(\Pi)$.
  
  - $Cn(\Pi)$ corresponds to the $\subseteq$-smallest model of $\Pi$
  
  - This is the set of consequences obtained by forward chaining.

- The set $Cn(\Pi)$ is an **answer set** of a **positive** program $\Pi$. 
Answer Set: Formal Definition

Positive programs

- A set of atoms $X$ is closed under a positive program $\Pi$ iff for any $r \in \Pi$: if $\text{body}^+(r) \subseteq X$ then $\text{head}(r) \in X$.
  
  $X$ corresponds to a model of $\Pi$ (seen as a formula).

- The smallest set of atoms which is closed under a positive program $\Pi$ is denoted by $\text{Cn}(\Pi)$.
  
  - $\text{Cn}(\Pi)$ corresponds to the $\subseteq$-smallest model of $\Pi$
  - This is the set of consequences obtained by forward chaining.

- The set $\text{Cn}(\Pi)$ is an answer set of a positive program $\Pi$.

Example

$$\{p \leftarrow, \quad q \leftarrow p, \quad r \leftarrow p, q, \quad t \leftarrow s\}$$

has answer set $\{p, q, r\}$. 
Answer set: Formal Definition

Normal programs
Answer set: Formal Definition

Normal programs

• Given a program \( \Pi \) and a set of atoms \( X \), the reduct, \( \Pi^X \), of \( \Pi \) relative to \( X \) is defined by

\[
\Pi^X = \{ head(r) \leftarrow body^+(r) \mid r \in \Pi \text{ and } body^-(r) \cap X = \emptyset \}.
\]

• Think of \( X \) as being a “guess” of an answer set.
• The reduct “compiles out” negation as failure, given \( X \).
Answer set: Formal Definition

Normal programs

• Given a program $\Pi$ and a set of atoms $X$, the reduct, $\Pi^X$, of $\Pi$ relative to $X$ is defined by

$$
\Pi^X = \{ head(r) \leftarrow body^+(r) \mid r \in \Pi \text{ and } body^-(r) \cap X = \emptyset \}.
$$

• Think of $X$ as being a “guess” of an answer set.
• The reduct “compiles out” negation as failure, given $X$.

• A set $X$ of atoms is an answer set of a program $\Pi$ if $Cn(\Pi^X) = X$. 

Answer set: Formal Definition

Normal programs

- Given a program $\Pi$ and a set of atoms $X$, the reduct, $\Pi^X$, of $\Pi$ relative to $X$ is defined by

$$\Pi^X = \{\text{head}(r) \leftarrow \text{body}^+(r) \mid r \in \Pi \text{ and } \text{body}^-(r) \cap X = \emptyset\}.$$  

- Think of $X$ as being a “guess” of an answer set.
- The reduct “compiles out” negation as failure, given $X$.

- A set $X$ of atoms is an answer set of a program $\Pi$ if $Cn(\Pi^X) = X$.

Recall: $Cn(\Pi^X)$ is the $\subseteq$-smallest (classical) model of $\Pi^X$. 
Answer set: Formal Definition

Normal programs

• Given a program $\Pi$ and a set of atoms $X$, the reduct, $\Pi^X$, of $\Pi$ relative to $X$ is defined by

$$\Pi^X = \{ head(r) \leftarrow body^+(r) \mid \ r \in \Pi \text{ and } body^-(r) \cap X = \emptyset \}.$$

• Think of $X$ as being a “guess” of an answer set.
• The reduct “compiles out” negation as failure, given $X$.

• A set $X$ of atoms is an answer set of a program $\Pi$ if $Cn(\Pi^X) = X$.

Recall: $Cn(\Pi^X)$ is the $\subseteq$–smallest (classical) model of $\Pi^X$.

Intuition: Every atom in $X$ is justified by an “applying rule” from $\Pi$
A Closer Look at $\Pi^X$

Given a set of atoms $X$ from $\Pi$, $\Pi^X$ is obtained from $\Pi$ by

1. deleting each rule having a $\text{not } A$ in its body with $A \in X$
   and then
2. deleting all negative atoms of the form $\text{not } A$ in the bodies of
   the remaining rules.

- Thus $\Pi^X$ is $\Pi$, but where negative atoms are taken into
  account.
- Then $X$ is an answer set of $\Pi$ just if $\Pi^X$ “generates” $X$, i.e.
  $Cn(\Pi^X) = X$. 
A first example

$$\Pi = \{ \ p \leftarrow p, \ q \leftarrow \text{not } p \}$$
A first example

\[ \Pi = \{ p \leftarrow p, \ q \leftarrow \text{not } p \} \]

<table>
<thead>
<tr>
<th>(X)</th>
<th>(\Pi^X)</th>
<th>(Cn(\Pi^X))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset)</td>
<td>(p \leftarrow p, \ q \leftarrow)</td>
<td>({q})</td>
</tr>
<tr>
<td>({p})</td>
<td>(p \leftarrow p)</td>
<td>(\emptyset)</td>
</tr>
<tr>
<td>({q})</td>
<td>(p \leftarrow p, \ q \leftarrow)</td>
<td>({q})</td>
</tr>
<tr>
<td>({p, q})</td>
<td>(p \leftarrow p)</td>
<td>(\emptyset)</td>
</tr>
</tbody>
</table>
A first example

\[ \Pi = \{ \ p \leftarrow p, \ q \leftarrow \text{not} \ p \} \]

<table>
<thead>
<tr>
<th>(X)</th>
<th>(\Pi^X)</th>
<th>(Cn(\Pi^X))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset)</td>
<td>(p \leftarrow p) \ \quad \quad \ \quad \quad q \leftarrow \</td>
<td></td>
</tr>
<tr>
<td>{p}</td>
<td>(p \leftarrow p)</td>
<td>(\emptyset)</td>
</tr>
<tr>
<td>{q}</td>
<td>(p \leftarrow p) \ \quad \quad \ \quad \quad {q}</td>
<td></td>
</tr>
<tr>
<td>{p, q}</td>
<td>(p \leftarrow p)</td>
<td>(\emptyset)</td>
</tr>
</tbody>
</table>
A first example

\[ \Pi = \{ p \leftarrow p, \quad q \leftarrow \text{not } p \} \]

<table>
<thead>
<tr>
<th>( \mathcal{X} )</th>
<th>( \Pi^\mathcal{X} )</th>
<th>( \text{Cn}(\Pi^\mathcal{X}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td>( p \leftarrow p )</td>
<td>( {q} \times )</td>
</tr>
<tr>
<td>{p}</td>
<td>( p \leftarrow p )</td>
<td>( \emptyset \times )</td>
</tr>
<tr>
<td>{q}</td>
<td>( p \leftarrow p )</td>
<td>( {q} )</td>
</tr>
<tr>
<td>{p, q}</td>
<td>( p \leftarrow p )</td>
<td>( \emptyset )</td>
</tr>
</tbody>
</table>
A first example

\[ \Pi = \{ p \leftarrow p, \; q \leftarrow \text{not } p \} \]

<table>
<thead>
<tr>
<th>(X)</th>
<th>(\Pi^X)</th>
<th>(Cn(\Pi^X))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset)</td>
<td>(p \leftarrow p)</td>
<td>({q})</td>
</tr>
<tr>
<td>({p})</td>
<td>(p \leftarrow p)</td>
<td>(\emptyset)</td>
</tr>
<tr>
<td>({q})</td>
<td>(p \leftarrow p)</td>
<td>({q})</td>
</tr>
<tr>
<td>({p, q})</td>
<td>(p \leftarrow p)</td>
<td>(\emptyset)</td>
</tr>
</tbody>
</table>
A first example

\[ \Pi = \{ \ p \leftarrow p, \ q \leftarrow \text{not } p \} \]

<table>
<thead>
<tr>
<th>(X)</th>
<th>(\Pi^X)</th>
<th>(Cn(\Pi^X))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset)</td>
<td>(p \leftarrow p) \quad (q \leftarrow)</td>
<td>({q}) (\times)</td>
</tr>
<tr>
<td>({p})</td>
<td>(p \leftarrow p) \quad (q \leftarrow)</td>
<td>(\emptyset) (\times)</td>
</tr>
<tr>
<td>({q})</td>
<td>(p \leftarrow p) \quad (q \leftarrow)</td>
<td>({q}) (\checkmark)</td>
</tr>
<tr>
<td>({p, q})</td>
<td>(p \leftarrow p) \quad (q \leftarrow)</td>
<td>(\emptyset) (\times)</td>
</tr>
</tbody>
</table>
A second example

\[ \Pi = \{ p \leftarrow \text{not } q, \quad q \leftarrow \text{not } p \} \]
A second example

\[ \Pi = \{ \ p \leftarrow \text{not } q, \ q \leftarrow \text{not } p \ \} \]

<table>
<thead>
<tr>
<th>(X)</th>
<th>(\Pi^X)</th>
<th>(Cn(\Pi^X))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset)</td>
<td>(p \leftarrow)</td>
<td>({p, q})</td>
</tr>
<tr>
<td>({p})</td>
<td>(p \leftarrow)</td>
<td>({p})</td>
</tr>
<tr>
<td>({q})</td>
<td></td>
<td>({q})</td>
</tr>
<tr>
<td>({p, q})</td>
<td>(q \leftarrow)</td>
<td>(\emptyset)</td>
</tr>
</tbody>
</table>
A second example

\[ \Pi = \{ p \leftarrow \text{not } q, \quad q \leftarrow \text{not } p \} \]

<table>
<thead>
<tr>
<th>(X)</th>
<th>(\Pi^X)</th>
<th>(\text{Cn}(\Pi^X))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset)</td>
<td>(p \leftarrow)</td>
<td>({p, q}) (\text{X})</td>
</tr>
<tr>
<td>({p})</td>
<td>(p \leftarrow)</td>
<td>({p})</td>
</tr>
<tr>
<td>({q})</td>
<td>(q \leftarrow)</td>
<td>({q})</td>
</tr>
<tr>
<td>({p, q})</td>
<td>(q \leftarrow)</td>
<td>(\emptyset)</td>
</tr>
</tbody>
</table>
A second example

\[ \Pi = \{ p \leftarrow \text{not } q, \quad q \leftarrow \text{not } p \} \]

<table>
<thead>
<tr>
<th>( X )</th>
<th>( \Pi^X )</th>
<th>( \text{Cn}(\Pi^X) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td>( p \leftarrow )</td>
<td>( {p, q} )</td>
</tr>
<tr>
<td></td>
<td>( q \leftarrow )</td>
<td></td>
</tr>
<tr>
<td>{p}</td>
<td>( p \leftarrow )</td>
<td>{p}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{q}</td>
<td></td>
<td>{q}</td>
</tr>
<tr>
<td></td>
<td>( q \leftarrow )</td>
<td></td>
</tr>
<tr>
<td>{p, q}</td>
<td></td>
<td>( \emptyset )</td>
</tr>
</tbody>
</table>
A second example

\[ \Pi = \{ \ p \leftarrow \text{not} \ q, \ q \leftarrow \text{not} \ p \ \} \]

<table>
<thead>
<tr>
<th>(X)</th>
<th>(\Pi^X)</th>
<th>(Cn(\Pi^X))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset)</td>
<td>(p \leftarrow)</td>
<td>({p, q})</td>
</tr>
<tr>
<td></td>
<td>(q \leftarrow)</td>
<td>(\times)</td>
</tr>
<tr>
<td>({p})</td>
<td>(p \leftarrow)</td>
<td>({p})</td>
</tr>
<tr>
<td></td>
<td>(q \leftarrow)</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>({q})</td>
<td>(q \leftarrow)</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>({p, q})</td>
<td></td>
<td>(\emptyset)</td>
</tr>
</tbody>
</table>
A second example

\[ \Pi = \{ \ p \leftarrow \neg q, \ q \leftarrow \neg p \} \]

<table>
<thead>
<tr>
<th>(X)</th>
<th>(\Pi^X)</th>
<th>(Cn(\Pi^X))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset)</td>
<td>(p \leftarrow)</td>
<td>{p, q}</td>
</tr>
<tr>
<td></td>
<td>(q \leftarrow)</td>
<td></td>
</tr>
<tr>
<td>{p}</td>
<td>(p \leftarrow)</td>
<td>{p}</td>
</tr>
<tr>
<td></td>
<td>(q \leftarrow)</td>
<td></td>
</tr>
<tr>
<td>{q}</td>
<td></td>
<td>{q}</td>
</tr>
<tr>
<td></td>
<td>(q \leftarrow)</td>
<td></td>
</tr>
<tr>
<td>{p, q}</td>
<td></td>
<td>\emptyset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A third example

$$\Pi = \{ p \leftarrow not \; p \}$$
A third example

\[
\Pi = \{ p \leftarrow \text{not } p \}
\]

<table>
<thead>
<tr>
<th>(X)</th>
<th>(\Pi^X)</th>
<th>(Cn(\Pi^X))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset)</td>
<td>(p \leftarrow)</td>
<td>({p})</td>
</tr>
<tr>
<td>({p})</td>
<td>(\emptyset)</td>
<td>(\emptyset)</td>
</tr>
</tbody>
</table>
A third example

$$\Pi = \{ \ p \leftarrow \text{not } p \ \}$$

<table>
<thead>
<tr>
<th>$X$</th>
<th>$\Pi^X$</th>
<th>$Cn(\Pi^X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>$p \leftarrow$</td>
<td>${p}$</td>
</tr>
<tr>
<td>${p}$</td>
<td></td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>
A third example

\[ \Pi = \{ p \leftarrow \text{not } p \} \]

<table>
<thead>
<tr>
<th>( X )</th>
<th>( \Pi^X )</th>
<th>( Cn(\Pi^X) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td>( p \leftarrow )</td>
<td>( {p} ) ✗</td>
</tr>
<tr>
<td>( {p} )</td>
<td>( )</td>
<td>( \emptyset ) ✗</td>
</tr>
</tbody>
</table>
A final example

$$\Pi = \{ a \leftarrow, \quad c \leftarrow \text{not } b, \text{not } d, \quad d \leftarrow a, \text{not } c, \}$$
A final example

$\Pi = \{ a \leftarrow, \ c \leftarrow \text{not } b, \text{not } d, \ d \leftarrow a, \text{not } c, \}$

This program has two answer sets, \( \{a, c\} \) and \( \{a, d\} \).
A final example

\[ \Pi = \{ \ a \leftarrow, \ c \leftarrow \text{not } b, \text{not } d, \ d \leftarrow a, \text{not } c, \ \} \]

This program has two answer sets, \{a, c\} and \{a, d\}. Here are 3 possibilities for \( X \):

<table>
<thead>
<tr>
<th>( X )</th>
<th>( \Pi^X )</th>
<th>( Cn(\Pi^X) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td>( a \leftarrow )</td>
<td>{a, c, d}</td>
</tr>
<tr>
<td></td>
<td>( c \leftarrow )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( d \leftarrow a )</td>
<td></td>
</tr>
<tr>
<td>{a, c}</td>
<td>( a \leftarrow )</td>
<td>{a, c}</td>
</tr>
<tr>
<td></td>
<td>( c \leftarrow )</td>
<td></td>
</tr>
<tr>
<td>{a, b, c, d}</td>
<td>( a \leftarrow )</td>
<td>{a}</td>
</tr>
</tbody>
</table>
A final example

\[ \Pi = \{ a \leftarrow, \ c \leftarrow \text{not } b, \text{not } d, \ d \leftarrow a, \text{not } c, \} \]

This program has two answer sets, \{a, c\} and \{a, d\}. Here are 3 possibilities for \( X \):

<table>
<thead>
<tr>
<th>( X )</th>
<th>( \Pi^X )</th>
<th>( Cn(\Pi^X) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td>( a \leftarrow )</td>
<td>( {a, c, d} ) ( \times )</td>
</tr>
<tr>
<td></td>
<td>( c \leftarrow )</td>
<td>( {a, c, d} ) ( \times )</td>
</tr>
<tr>
<td></td>
<td>( d \leftarrow a )</td>
<td>( {a, c, d} ) ( \times )</td>
</tr>
<tr>
<td>{a, c}</td>
<td>( a \leftarrow )</td>
<td>( {a, c} )</td>
</tr>
<tr>
<td></td>
<td>( c \leftarrow )</td>
<td>( {a, c} )</td>
</tr>
<tr>
<td>{a, b, c, d}</td>
<td>( a \leftarrow )</td>
<td>( {a} )</td>
</tr>
</tbody>
</table>
A final example

\[ \Pi = \{ \ a \leftarrow, \ c \leftarrow \text{not} \ b, \text{not} \ d, \ d \leftarrow a, \text{not} \ c, \ \} \]

This program has two answer sets, \( \{a, c\} \) and \( \{a, d\} \).

Here are 3 possibilities for \( X \):

<table>
<thead>
<tr>
<th>( X )</th>
<th>( \Pi^X )</th>
<th>( \text{Cn}(\Pi^X) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td>( a \leftarrow )</td>
<td>( {a, c, d} )  ( \times )</td>
</tr>
<tr>
<td></td>
<td>( c \leftarrow )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>( d \leftarrow a )</td>
<td></td>
</tr>
<tr>
<td>( {a, c} )</td>
<td>( a \leftarrow )</td>
<td>( {a, c} )  ( \checkmark )</td>
</tr>
<tr>
<td></td>
<td>( c \leftarrow )</td>
<td></td>
</tr>
<tr>
<td>( {a, b, c, d} )</td>
<td>( a \leftarrow )</td>
<td>( {a} )</td>
</tr>
</tbody>
</table>


A final example

\[ \Pi = \{ \ a \leftarrow, \ c \leftarrow \text{not } b, \text{not } d, \ d \leftarrow a, \text{not } c, \ \} \]

This program has two answer sets, \{a, c\} and \{a, d\}. Here are 3 possibilities for \(X\):

<table>
<thead>
<tr>
<th>(X)</th>
<th>(\Pi^X)</th>
<th>(\text{Cn}(\Pi^X))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset)</td>
<td>(a \leftarrow)</td>
<td>({a, c, d})</td>
</tr>
<tr>
<td></td>
<td>(c \leftarrow)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(d \leftarrow a)</td>
<td></td>
</tr>
<tr>
<td>{a, c}</td>
<td>(a \leftarrow)</td>
<td>({a, c})</td>
</tr>
<tr>
<td></td>
<td>(c \leftarrow)</td>
<td></td>
</tr>
<tr>
<td>{a, b, c, d}</td>
<td>(a \leftarrow)</td>
<td>({a})</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(X)</th>
<th>(\Pi^X)</th>
<th>(\text{Cn}(\Pi^X))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\emptyset)</td>
<td>(a \leftarrow)</td>
<td>({a, c, d})</td>
</tr>
<tr>
<td></td>
<td>(c \leftarrow)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(d \leftarrow a)</td>
<td></td>
</tr>
<tr>
<td>{a, c}</td>
<td>(a \leftarrow)</td>
<td>({a, c})</td>
</tr>
<tr>
<td></td>
<td>(c \leftarrow)</td>
<td></td>
</tr>
<tr>
<td>{a, b, c, d}</td>
<td>(a \leftarrow)</td>
<td>({a})</td>
</tr>
</tbody>
</table>
Answer sets: Some properties

- A program may have zero, one, or multiple answer sets.
- If $X$ is an answer set of a logic program $\Pi$, then $X$ is a model of $\Pi$ (seen as formulas of classical logic).
- If $X$ and $Y$ are answer sets of a logic program $\Pi$, then $X \not\subset Y$. 
Let $\Pi$ be a logic program.

- The *Herbrand Universe* $U^\Pi$ is the set of constants in $\Pi$
- The *Herbrand Base* $B^\Pi$ is the set of (variable-free) atoms constructible from $U^\Pi$

We usually denote this as $\mathcal{A}$, and call it the *alphabet*.
Programs with Variables

- **Ground instances** of $r \in \Pi$:

  Set of variable-free rules obtained by replacing all variables in $r$ by elements from $U^\Pi$:

  $$\text{ground}(r) = \{ r\theta \mid \theta : \text{var}(r) \rightarrow U^\Pi \}$$

  where $\text{var}(r)$ stands for the set of all variables occurring in $r$ and $\theta$ is a (ground) substitution.
Programs with Variables

• *Ground instances* of $r \in \Pi$:

Set of variable-free rules obtained by replacing all variables in $r$ by elements from $U^\Pi$:

$$\text{ground}(r) = \{ r\theta \mid \theta : \text{var}(r) \rightarrow U^\Pi \}$$

where $\text{var}(r)$ stands for the set of all variables occurring in $r$ and $\theta$ is a (ground) substitution.

• *Ground instantiation* of $\Pi$:

$$\text{ground}(\Pi) = \{ \text{ground}(r) \mid r \in \Pi \}$$
An Example

\[ \Pi = \{ \ r(a, b) \leftarrow, \ r(b, c) \leftarrow, \ t(X, Y) \leftarrow r(X, Y) \ \} \]
An Example

\[ \Pi = \{ r(a, b) \leftarrow, \quad r(b, c) \leftarrow, \quad t(X, Y) \leftarrow r(X, Y) \} \]

\[ U^\Pi = \{ a, b, c \} \]
An Example

\[ \Pi = \{ \ r(a, b) \leftarrow, \ r(b, c) \leftarrow, \ t(X, Y) \leftarrow r(X, Y) \ \} \]

\[ U^\Pi = \{ a, b, c \} \]

\[ B^\Pi = \{ \begin{aligned} r(a, a), & \ r(a, b), & \ r(a, c), \\ r(b, a), & \ r(b, b), & \ r(b, c), \\ r(c, a), & \ r(c, b), & \ r(c, c), \\ t(a, a), & \ t(a, b), & \ t(a, c), \\ t(b, a), & \ t(b, b), & \ t(b, c), \\ t(c, a), & \ t(c, b), & \ t(c, c) \end{aligned} \} \]
An Example

\[ \Pi = \{ r(a, b) \leftarrow, \ r(b, c) \leftarrow, \ t(X, Y) \leftarrow r(X, Y) \} \]

\[ U^\Pi = \{ a, b, c \} \]

\[ B^\Pi = \begin{cases} 
  r(a, a), \ r(a, b), \ r(a, c), \\
  r(b, a), \ r(b, b), \ r(b, c), \\
  r(c, a), \ r(c, b), \ r(c, c), \\
  t(a, a), \ t(a, b), \ t(a, c), \\
  t(b, a), \ t(b, b), \ t(b, c), \\
  t(c, a), \ t(c, b), \ t(c, c) \end{cases} \]

\[ ground(\Pi) = \begin{cases} 
  r(a, b) \leftarrow, \\
  r(b, c) \leftarrow, \\
  t(a, a) \leftarrow r(a, a), \ t(b, a) \leftarrow r(b, a), \ t(c, a) \leftarrow r(c, a), \\
  t(a, b) \leftarrow r(a, b), \ t(b, b) \leftarrow r(b, b), \ t(c, b) \leftarrow r(c, b), \\
  t(a, c) \leftarrow r(a, c), \ t(b, c) \leftarrow r(b, c), \ t(c, c) \leftarrow r(c, c) \end{cases} \]
An Example

\[ \Pi = \{ \quad r(a, b) \leftarrow, \quad r(b, c) \leftarrow, \quad t(X, Y) \leftarrow r(X, Y) \} \]

\[ U^\Pi = \{ a, b, c \} \]

\[ B^\Pi = \begin{cases} 
   r(a, a), \ r(a, b), \ r(a, c), \\
   r(b, a), \ r(b, b), \ r(b, c), \\
   r(c, a), \ r(c, b), \ r(c, c), \\
   t(a, a), \ t(a, b), \ t(a, c), \\
   t(b, a), \ t(b, b), \ t(b, c), \\
   t(c, a), \ t(c, b), \ t(c, c) 
\end{cases} \]

\[ \text{ground}(\Pi) = \begin{cases} 
   r(a, b) \leftarrow, \\
   r(b, c) \leftarrow, \\
   t(a, b) \leftarrow, \\
   t(b, c) \leftarrow 
\end{cases} \]

• *Intelligent Grounding* aims to reduce the ground instantiation.
Answer Sets of Programs with Variables

Let $\Pi$ be a normal logic program with variables.

We define a set $X$ of \textit{(ground)} atoms as an \textit{answer set} of $\Pi$ if $\text{Cn}(\text{ground}(\Pi)^X) = X$. 
Programs with Integrity Constraints

**Purpose:** Integrity constraints eliminate unwanted candidate solutions
Programs with Integrity Constraints

**Purpose:** Integrity constraints eliminate unwanted candidate solutions

**Syntax:** An integrity constraint is of the form

\[
\leftarrow A_1, \ldots, A_m, \text{not } A_{m+1}, \ldots, \text{not } A_n,
\]

where \( n \geq m \geq 1 \), and each \( A_i \) (\( 1 \leq i \leq n \)) is a atom.

**Example**

\[
\leftarrow \text{Edge}(X, Y), \text{Col}(X, C), \text{Col}(Y, C)
\]
Programs with Integrity Constraints

Purpose: Integrity constraints eliminate unwanted candidate solutions

Syntax: An integrity constraint is of the form

\[ \leftarrow A_1, \ldots, A_m, \text{not } A_{m+1}, \ldots, \text{not } A_n, \]

where \( n \geq m \geq 1 \), and each \( A_i \) (\( 1 \leq i \leq n \)) is a atom.

Example

\[ \leftarrow \text{Edge}(X, Y), \text{Col}(X, C), \text{Col}(Y, C) \]

Implementation: For a new symbol \( x \),

\[ \text{map: } \leftarrow A_1, \ldots, A_m, \text{not } A_{m+1}, \ldots, \text{not } A_n \]
\[ \text{to: } x \leftarrow A_1, \ldots, A_m, \text{not } A_{m+1}, \ldots, \text{not } A_n, \text{not } x \]
Computation: Standard Approach

Global parameters: Logic program $\Pi$ and its set of atoms $A$.

- $X$ is a set of atoms known to be true;
- $Y$ is a set of atoms known to be false.
- Initially $X = Y = \emptyset$.

$\text{answerset } \Pi(X, Y)$:
1. $(X, Y) \leftarrow \text{propagation } \Pi(X, Y)$
2. If $(X \cap Y) \neq \emptyset$ then fail
3. If $(X \cup Y) = A$ then return $(X)$
4. Select $A \in A \setminus (X \cup Y)$
5. $\text{answerset } \Pi(X \cup \{A\}, Y)$
6. $\text{answerset } \Pi(X, Y \cup \{A\})$
Computation: Standard Approach

Global parameters: Logic program Π and its set of atoms A.

- $X$ is a set of atoms known to be true;
- $Y$ is a set of atoms known to be false.
- Initially $X = Y = \emptyset$.
Computation: Standard Approach

Global parameters: Logic program \( \Pi \) and its set of atoms \( \mathcal{A} \).

- \( X \) is a set of atoms known to be true;
- \( Y \) is a set of atoms known to be false.
- Initially \( X = Y = \emptyset \).

\[
\text{answerset}_\Pi(X, Y) : \\
1. (X, Y) \leftarrow \text{propagation}_\Pi(X, Y) \\
2. \text{if } (X \cap Y) \neq \emptyset \text{ then fail} \\
3. \text{if } (X \cup Y) = \mathcal{A} \text{ then return}(X) \\
4. \text{select } A \in \mathcal{A} \setminus (X \cup Y) \\
5. \text{answerset}_\Pi(X \cup \{A\}, Y) \\
6. \text{answerset}_\Pi(X, Y \cup \{A\})
\]
Computation: Standard Approach

Comments:
• \((X, Y)\) is supposed to be a 3-valued model such that
  • \(X \subseteq Z\) and
  • \(Y \cap Z = \emptyset\)

for an answer set \(Z\) of \(\Pi\).
Computation: Standard Approach

Comments:

- \((X, Y)\) is supposed to be a 3-valued model such that
  - \(X \subseteq Z\) and
  - \(Y \cap Z = \emptyset\)

for an answer set \(Z\) of \(\Pi\).

- Key operations:
  - \(\text{propagation}_{\Pi}(X, Y)\) and
  - "\(\text{select } A \in A \setminus (X \cup Y)\)"

Worst case complexity: \(O(2^{|A|})\)

More later...
Computation: Standard Approach

Comments:

- \((X, Y)\) is supposed to be a 3-valued model such that
  - \(X \subseteq Z\) and
  - \(Y \cap Z = \emptyset\)

  for an answer set \(Z\) of \(\Pi\).

- Key operations:
  - \(\text{propagation}_{\Pi}(X, Y)\) and
  - “\(\text{select } A \in A \setminus (X \cup Y)\)”

- Worst case complexity: \(O(2^{|A|})\)

More later...