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

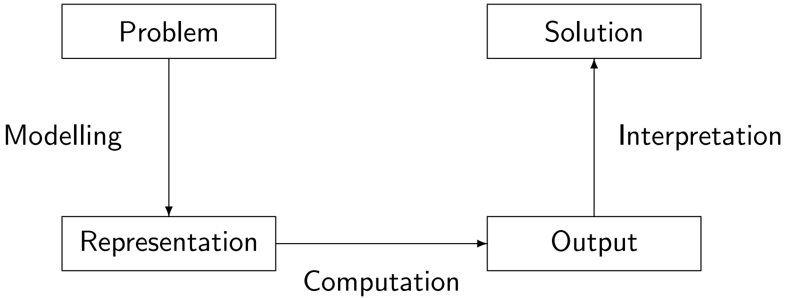
Goal: Declarative problem solving

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

C(I)	vertex(1)	$\leftarrow$	edge(1,2)	$\leftarrow$
	vertex(2)	$\leftarrow$	edge(2,3)	$\leftarrow$
	vertex(3)	$\leftarrow$	edge(3,1)	$\leftarrow$
C(P)	coloured(V,r)	$\leftarrow$	not coloured(V,b), not coloured(V,g), vertex(V)	
	coloured(V,b)	$\leftarrow$	not coloured(V,r), not coloured(V,g), vertex(V)	
	coloured(V,g)	$\leftarrow$	not coloured(V,r), not coloured(V,b), vertex(V)	
		$\leftarrow$	edge(V,U), coloured(V,C), coloured(U,C), colour(C)	
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

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

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

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

ASP: Idea

- A (normal) rule, r, is of the form

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

  - *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.

ASP: Idea

- A (normal) rule, r, is of the form

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

  - *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. $\{a \leftarrow b, not\ c.,\ \ b.\}$ has answer set $\{a, b\}$.

ASP: Idea

- A (normal) rule, r, is of the form

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

  - *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. $\{a \leftarrow b, not\ c., \quad b.\}$ has answer set $\{a, b\}$.
  $\{a \leftarrow not\ b., \quad b \leftarrow not\ a.\}$ 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: $answer(42)$, $coloured(1, red)$, hot
- An atom can be either true or false

Terms

- Terms are the subatomic components of atoms
- Terms represent objects
    - Examples: 42, red, joe
- Variables are also terms, but are removed from a program by 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

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

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

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

where $n, m \geq 0$, and each A_i is an atom.

- Notation

$$\begin{array}{rcl} \mathit{head}(r) & = & A_0 \\ \mathit{body}(r) & = & \{A_1, \ldots, A_m, \mathit{not}\ A_{m+1}, \ldots, \mathit{not}\ A_n\} \\ \mathit{body}^+(r) & = & \{A_1, \ldots, A_m\} \\ \mathit{body}^-(r) & = & \{A_{m+1}, \ldots, A_n\} \end{array}$$

Normal logic programs

- A (normal) logic program is a finite set of rules.
- A (normal) rule, r, is of the form

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

  where $n, m \geq 0$, and each A_i is an atom.
- Notation

$$\begin{aligned} head(r) &= A_0 \\ body(r) &= \{A_1, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n\} \\ body^+(r) &= \{A_1, \ldots, A_m\} \\ body^-(r) &= \{A_{m+1}, \ldots, A_n\} \end{aligned}$$

- A program is called positive if $body^-(r) = \emptyset$ for all its rules.
  - ☞ $=$ 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:

	if	and	or	negation as failure	classical negation
logic program	$\leftarrow$	,	;	*not*/$\sim$	$\neg$
formula	$\rightarrow$	$\wedge$	$\vee$		$\neg$
source code	`:-`	`,`	`|`	`not`	`-`

Answer Set: Intuitions

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

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

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.

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 *true* and
  - atoms not in X are *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 *true* and
  - atoms not in X are *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 *true* and
  - atoms not in X are *false*.
- Program $\{a \leftarrow \mathit{not}\ b\}$ has answer set $\{a\}$.

☞ The negation-as-failure operator *not* makes a difference!

Answer sets: Basic idea

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

Answer sets: Basic idea

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

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

Answer sets: Basic idea

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

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

The corresponding logic program is:

$$
\begin{array}{lll}
q & \leftarrow & \\
p & \leftarrow & q,\ \mathit{not}\ r
\end{array}
$$

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:

$$
\begin{array}{lll}
q & \leftarrow & \\
p & \leftarrow & q,\ \mathit{not}\ r
\end{array}
$$

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

Answer sets: Basic idea

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

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

The corresponding logic program is:

$$
\begin{array}{lll}
q & \leftarrow & \\
p & \leftarrow & q, \ \mathit{not}\ r
\end{array}
$$

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

Roughly, a set of atoms X is an answer set of a logic program Π if

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

☞ 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 Π iff for any $r \in \Pi$: if $body^+(r) \subseteq X$ then $head(r) \in X$.
  - ☞ X corresponds to a model of Π (seen as a formula).

Answer Set: Formal Definition

Positive programs

- A set of atoms X is closed under a positive program Π iff for any $r \in \Pi$: if $body^+(r) \subseteq X$ then $head(r) \in X$.
  - ☞ X corresponds to a model of Π (seen as a formula).
- The smallest set of atoms which is closed under a positive program Π is denoted by $Cn(\Pi)$.
  - $Cn(\Pi)$ corresponds to the $\subseteq$-smallest model of Π
  - 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 Π iff for any $r \in \Pi$: if $body^+(r) \subseteq X$ then $head(r) \in X$.
  - ☞ X corresponds to a model of Π (seen as a formula).
- The smallest set of atoms which is closed under a positive program Π is denoted by $Cn(\Pi)$.
  - $Cn(\Pi)$ corresponds to the $\subseteq$-smallest model of Π
  - This is the set of consequences obtained by forward chaining.
- The set $Cn(\Pi)$ is an answer set of a *positive* program Π.

Answer Set: Formal Definition

Positive programs

- A set of atoms X is closed under a positive program Π iff for any $r \in \Pi$: if $body^+(r) \subseteq X$ then $head(r) \in X$.
  - ☞ X corresponds to a model of Π (seen as a formula).
- The smallest set of atoms which is closed under a positive program Π is denoted by $Cn(\Pi)$.
  - $Cn(\Pi)$ corresponds to the $\subseteq$-smallest model of Π
  - This is the set of consequences obtained by forward chaining.
- The set $Cn(\Pi)$ is an answer set of a *positive* program Π.

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 Π and a set of atoms X, the reduct, Π^X, of Π 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 Π and a set of atoms X,
  the reduct, Π^X, of Π 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 Π if
  $Cn(\Pi^X) = X$.

Answer set: Formal Definition

Normal programs

- Given a program Π and a set of atoms X,
  the reduct, Π^X, of Π 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 Π if
  $Cn(\Pi^X) = X$.

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

Answer set: Formal Definition

Normal programs

- Given a program Π and a set of atoms X,
  the reduct, Π^X, of Π 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 Π if
  $Cn(\Pi^X) = X$.

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

Intuition: Every atom in X is justified by an *"applying rule" from Π*

A Closer Look at Π^X

Given a set of atoms X from Π, Π^X is obtained from Π by

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

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

A first example

$$\Pi = \{\ p \leftarrow p, \quad q \leftarrow not\ p\ \}$$

A first example

$\Pi = \{ p \leftarrow p, \quad q \leftarrow not\ p \}$

X	Π^X	$Cn(\Pi^X)$
$\emptyset$	$p \leftarrow p$ $q \leftarrow$	$\{q\}$
$\{p\}$	$p \leftarrow p$	$\emptyset$
$\{q\}$	$p \leftarrow p$ $q \leftarrow$	$\{q\}$
$\{p, q\}$	$p \leftarrow p$	$\emptyset$

A first example

$$\Pi = \{\ p \leftarrow p, \quad q \leftarrow not\ p\ \}$$

X	Π^X	$Cn(\Pi^X)$	
$\emptyset$	$p \leftarrow p$ $q \leftarrow$	$\{q\}$	✘
$\{p\}$	$p \leftarrow p$	$\emptyset$	
$\{q\}$	$p \leftarrow p$ $q \leftarrow$	$\{q\}$	
$\{p, q\}$	$p \leftarrow p$	$\emptyset$	

A first example

$\Pi = \{\ p \leftarrow p, \quad q \leftarrow not\ p\ \}$

X	Π^X	$Cn(\Pi^X)$	
$\emptyset$	$p \leftarrow p$ $q \leftarrow$	$\{q\}$	✘
$\{p\}$	$p \leftarrow p$	$\emptyset$	✘
$\{q\}$	$p \leftarrow p$ $q \leftarrow$	$\{q\}$	
$\{p, q\}$	$p \leftarrow p$	$\emptyset$	

A first example

$$\Pi = \{\ p \leftarrow p, \quad q \leftarrow not\ p\ \}$$

X	Π^X	$Cn(\Pi^X)$	
$\emptyset$	$p \leftarrow p$ $q \leftarrow$	$\{q\}$	✘
$\{p\}$	$p \leftarrow p$	$\emptyset$	✘
$\{q\}$	$p \leftarrow p$ $q \leftarrow$	$\{q\}$	✔
$\{p, q\}$	$p \leftarrow p$	$\emptyset$	

A first example

$\Pi = \{ \; p \leftarrow p, \quad q \leftarrow not \;\; p \; \}$

X	Π^X	$Cn(\Pi^X)$	
$\emptyset$	$p \leftarrow p$ $q \leftarrow$	$\{q\}$	✘
$\{p\}$	$p \leftarrow p$	$\emptyset$	✘
$\{q\}$	$p \leftarrow p$ $q \leftarrow$	$\{q\}$	✔
$\{p, q\}$	$p \leftarrow p$	$\emptyset$	✘

A second example

$$\Pi = \{\ p \leftarrow \mathit{not}\ q, \quad q \leftarrow \mathit{not}\ p\ \}$$

A second example

$$\Pi = \{\ p \leftarrow not\ \ q, \quad q \leftarrow not\ \ p\ \}$$

X	Π^X	$Cn(\Pi^X)$
$\emptyset$	$p \leftarrow$ $q \leftarrow$	$\{p, q\}$
$\{p\}$	$p \leftarrow$	$\{p\}$
$\{q\}$	$q \leftarrow$	$\{q\}$
$\{p, q\}$		$\emptyset$

A second example

$$\Pi = \{\ p \leftarrow not\ q,\quad q \leftarrow not\ p\ \}$$

X	Π^X	$Cn(\Pi^X)$
$\emptyset$	$p \leftarrow$ $q \leftarrow$	$\{p, q\}$ ✘
$\{p\}$	$p \leftarrow$	$\{p\}$
$\{q\}$	$q \leftarrow$	$\{q\}$
$\{p, q\}$		$\emptyset$

A second example

$$\Pi = \{\ p \leftarrow \mathit{not}\ q, \quad q \leftarrow \mathit{not}\ p\ \}$$

X	Π^X	$Cn(\Pi^X)$	
$\emptyset$	$p \leftarrow$ $q \leftarrow$	$\{p, q\}$	✘
$\{p\}$	$p \leftarrow$	$\{p\}$	✔
$\{q\}$	$q \leftarrow$	$\{q\}$	
$\{p, q\}$		$\emptyset$	

A second example

$$\Pi = \{\, p \leftarrow not\ q, \quad q \leftarrow not\ p \,\}$$

X	Π^X	$Cn(\Pi^X)$	
$\emptyset$	$p \leftarrow$ $q \leftarrow$	$\{p, q\}$	✘
$\{p\}$	$p \leftarrow$	$\{p\}$	✔
$\{q\}$	$q \leftarrow$	$\{q\}$	✔
$\{p, q\}$		$\emptyset$	

A second example

$$\Pi = \{\ p \leftarrow not\ q, \quad q \leftarrow not\ p\ \}$$

X	Π^X	$Cn(\Pi^X)$	
$\emptyset$	$p \leftarrow$ $q \leftarrow$	$\{p, q\}$	✘
$\{p\}$	$p \leftarrow$	$\{p\}$	✔
$\{q\}$	$q \leftarrow$	$\{q\}$	✔
$\{p, q\}$		$\emptyset$	✘

A third example

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

A third example

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

X	Π^X	$Cn(\Pi^X)$
$\emptyset$	$p \quad \leftarrow$	$\{p\}$
$\{p\}$		$\emptyset$

A third example

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

X	Π^X	$Cn(\Pi^X)$
$\emptyset$	$p \quad \leftarrow$	$\{p\}$ ✘
$\{p\}$		$\emptyset$

A third example

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

X	Π^X	$Cn(\Pi^X)$	
$\emptyset$	$p \quad \leftarrow$	$\{p\}$	✘
$\{p\}$		$\emptyset$	✘

A final example

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

A final example

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

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

A final example

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

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

X	Π^X	$Cn(\Pi^X)$
$\emptyset$	$a \leftarrow$ $c \leftarrow$ $d \leftarrow a$	$\{a, c, d\}$
$\{a, c\}$	$a \leftarrow$ $c \leftarrow$	$\{a, c\}$
$\{a, b, c, d\}$	$a \leftarrow$	$\{a\}$

A final example

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

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

X	Π^X	$Cn(\Pi^X)$
$\emptyset$	$a \leftarrow$ $c \leftarrow$ $d \leftarrow a$	$\{a, c, d\}$ ✘
$\{a, c\}$	$a \leftarrow$ $c \leftarrow$	$\{a, c\}$
$\{a, b, c, d\}$	$a \leftarrow$	$\{a\}$

A final example

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

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

X	Π^X	$Cn(\Pi^X)$	
$\emptyset$	$a \leftarrow$ $c \leftarrow$ $d \leftarrow a$	$\{a, c, d\}$	✘
$\{a, c\}$	$a \leftarrow$ $c \leftarrow$	$\{a, c\}$	✔
$\{a, b, c, d\}$	$a \leftarrow$	$\{a\}$	

A final example

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

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

X	Π^X	$Cn(\Pi^X)$	
$\emptyset$	$a \leftarrow$ $c \leftarrow$ $d \leftarrow a$	$\{a, c, d\}$	✘
$\{a, c\}$	$a \leftarrow$ $c \leftarrow$	$\{a, c\}$	✔
$\{a, b, c, d\}$	$a \leftarrow$	$\{a\}$	✘

Answer sets: Some properties

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

Programs with Variables

Let Π be a logic program.

- The *Herbrand Universe* U^{Π} is the set of constants in Π
- The *Herbrand Base* B^{Π} is the set of (variable-free) atoms constructible from U^{Π}
  - ☞ 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^{Π}:

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

  where $var(r)$ stands for the set of all variables occurring in r and θ 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^{Π}:

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

  where $var(r)$ stands for the set of all variables occurring in r and θ is a (ground) substitution.

- *Ground instantiation* of Π:

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

An Example

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

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

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

$$U^{\Pi} = \{a, b, c\}$$

$$B^{\Pi} = \left\{ \begin{array}{c} r(a,a),\; r(a,b),\; r(a,c), \\ r(b,a),\; r(b,b),\; r(b,c), \\ r(c,a),\; r(c,b),\; r(c,c), \\ t(a,a),\; t(a,b),\; t(a,c), \\ t(b,a),\; t(b,b),\; t(b,c), \\ t(c,a),\; t(c,b),\; t(c,c) \end{array} \right\}$$

An Example

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

$$U^{\Pi} = \{a, b, c\}$$

$$B^{\Pi} = \left\{ \begin{array}{c} r(a,a),\ r(a,b),\ r(a,c), \\ r(b,a),\ r(b,b),\ r(b,c), \\ r(c,a),\ r(c,b),\ r(c,c), \\ t(a,a),\ t(a,b),\ t(a,c), \\ t(b,a),\ t(b,b),\ t(b,c), \\ t(c,a),\ t(c,b),\ t(c,c) \end{array} \right\}$$

$$ground(\Pi) = \left\{ \begin{array}{lllllll} r(a,b) & \leftarrow & , & & & & \\ r(b,c) & \leftarrow & , & & & & \\ t(a,a) & \leftarrow & r(a,a), & t(b,a) \leftarrow & r(b,a), & t(c,a) \leftarrow & r(c,a), \\ t(a,b) & \leftarrow & r(a,b), & t(b,b) \leftarrow & r(b,b), & t(c,b) \leftarrow & r(c,b), \\ t(a,c) & \leftarrow & r(a,c), & t(b,c) \leftarrow & r(b,c), & t(c,c) \leftarrow & r(c,c) \end{array} \right\}$$

An Example

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

$$U^{\Pi} = \{a, b, c\}$$

$$B^{\Pi} = \left\{ \begin{array}{c} r(a,a),\ r(a,b),\ r(a,c), \\ r(b,a),\ r(b,b),\ r(b,c), \\ r(c,a),\ r(c,b),\ r(c,c), \\ t(a,a),\ t(a,b),\ t(a,c), \\ t(b,a),\ t(b,b),\ t(b,c), \\ t(c,a),\ t(c,b),\ t(c,c) \end{array} \right\}$$

$$ground(\Pi) = \left\{ \begin{array}{ll} r(a,b) \leftarrow \ , & \\ r(b,c) \leftarrow \ , & \\ & \\ t(a,b) \leftarrow \qquad\qquad , & \\ & t(b,c) \leftarrow \qquad\qquad , \end{array} \right\}$$

- *Intelligent Grounding* aims to reduce the ground instantiation.

Answer Sets of Programs with Variables

Let Π be a normal logic program with variables.

We define a set X of (*ground*) atoms as an *answer set* of Π if $Cn(ground(\Pi)^X) = X$.

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

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

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

Example

$$\leftarrow Edge(X, Y), Col(X, C), Col(Y, C)$$

Programs with Integrity Constraints

Purpose: Integrity constraints eliminate unwanted candidate solutions

Syntax: An integrity constraint is of the form

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

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

Example

$$\leftarrow Edge(X, Y), Col(X, C), Col(Y, C)$$

Implementation: For a new symbol x,

map:		$\leftarrow$	$A_1, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n$
to:	x	$\leftarrow$	$A_1, \ldots, A_m, not\ A_{m+1}, \ldots, not\ A_n, not\ x$

Computation: Standard Approach

Computation: Standard Approach

Global parameters: Logic program Π 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 $X = Y = \emptyset$.

Computation: Standard Approach

Global parameters: Logic program Π 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 $X = Y = \emptyset$.

answerset$_\Pi(X, Y)$:

1. $(X, Y) \leftarrow$ *propagation*$_\Pi(X, Y)$
2. **if** $(X \cap Y) \neq \emptyset$ **then fail**
3. **if** $(X \cup Y) = \mathcal{A}$ **then return**(X)
4. **select** $A \in \mathcal{A} \setminus (X \cup Y)$
5. *answerset*$_\Pi(X \cup \{A\}, Y)$
6. *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 Π.

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 Π.
- Key operations:
  - $propagation_\Pi(X, Y)$ and
  - "**select** $A \in \mathcal{A} \setminus (X \cup Y)$"

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 Π.
- Key operations:
  - $\mathit{propagation}_{\Pi}(X, Y)$ and
  - "**select** $A \in \mathcal{A} \setminus (X \cup Y)$"
- Worst case complexity: $\mathcal{O}(2^{|\mathcal{A}|})$

☞ More later...