Answer Set Programming\textsuperscript{1}

Abdallah Saffidine

COMP4418

\textsuperscript{1}Slides designed by Christoph Schwering
Non-Monotonic Reasoning

∀x (Car(x) → ¬Entry(x))
∀x (Car(x) ∧ Auth(x) → Entry(x))

|= Car(C) ∧ Auth(C) → ¬Entry(C)
Non-Monotonic Reasoning

\[ \forall x \left( \text{Car}(x) \rightarrow \neg \text{Entry}(x) \right) \]

\[ \forall x \left( \text{Car}(x) \land \text{Auth}(x) \rightarrow \text{Entry}(x) \right) \]
Non-Monotonic Reasoning

∀x (Car(x) → ¬Entry(x))
∀x (Car(x) ∧ Auth(x) → Entry(x))
Non-Monotonic Reasoning

\[ \forall x (\text{Car}(x) \rightarrow \neg \text{Entry}(x)) \]
\[ \forall x (\text{Car}(x) \land \text{Auth}(x) \rightarrow \text{Entry}(x)) \]
\[ \models \text{Car}(C) \land \text{Auth}(C) \rightarrow \neg \text{Entry}(C) \]
ASP at a Glance

- **ASP = Answer Set Programming**
  - ASP ≠ Microsoft's Active Server Pages

- ASP belongs to logic programming
  - Like Prolog: $\text{Head} \leftarrow \text{Body}$ or $\text{Head} : - \text{Body}$.
  - Like Prolog: negation as failure
  - Unlike Prolog: $\text{Head}$ may be empty ⇒ constraints

- **Declarative programming**
  - Unlike Prolog: no procedural control
  - Order has no impact on semantics

- **ASP programs compute models**
  - Unlike Prolog: not query-oriented, no resolution
  - Unlike Prolog: not Turing-complete
  - Tool for problems in NP and NP$^\text{NP}$
Motivation for ASP and this Lecture

- Very useful in practice!
  - Declarative problem solving
  - Very fast to write
  - Very fast to run
  - Few experts

- Interesting case study
  - Small, simple core language
  - Great expressivity by reduction to core language

- Knowing the theory is essential
Example: Graph Colouring

Definition: graph colouring problem

Input: graph with vertices \( V \) and edges \( E \subseteq V \times V \), set of colors \( C \). Is there a mapping \( m : V \rightarrow C \) with \( m(x) \neq m(y) \) for all \( (x, y) \in E \)?
Definition: graph colouring problem

Input: graph with vertices $V$ and edges $E \subseteq V \times V$, set of colors $C$. Is there a mapping $m : V \rightarrow C$ with $m(x) \neq m(y)$ for all $(x, y) \in E$?
Example: Graph Colouring

Definition: graph colouring problem

Input: graph with vertices $V$ and edges $E \subseteq V \times V$, set of colors $C$. Is there a mapping $m : V \rightarrow C$ with $m(x) \neq m(y)$ for all $(x,y) \in E$?

Graph Colouring is NP-complete

- NP: guess solution, verify in polynomial time
- NP-complete: among hardest in NP

Many applications:

- Mapping (neighbouring countries to different colors)
- Compilers (register allocation)
- Scheduling (e.g., conflicting jobs to different time slots)
- Allocation problems, Sudoku, ...
Example: Graph Colouring

Definition: graph colouring problem

Input: graph with vertices $V$ and edges $E \subseteq V \times V$, set of colors $C$. Is there a mapping $m : V \rightarrow C$ with $m(x) \neq m(y)$ for all $(x,y) \in E$?
Example: Graph Colouring

Definition: graph colouring problem

Input: graph with vertices $V$ and edges $E \subseteq V \times V$, set of colors $C$. Is there a mapping $m : V \to C$ with $m(x) \neq m(y)$ for all $(x, y) \in E$?

Diagram:

- Vertex 1
- Vertex 2
- Vertex 3
- Vertex 4
- Vertex 5
- Vertex 6

Edges:
- $e(1, 2)$
- $e(1, 3)$
- $e(1, 4)$
- $e(2, 4)$
- $e(2, 5)$
- $e(2, 6)$
- $e(3, 1)$
- $e(3, 4)$
- $e(3, 5)$
- $e(4, 1)$
- $e(4, 2)$
- $e(5, 3)$
- $e(5, 4)$
- $e(5, 6)$
- $e(6, 2)$
- $e(6, 3)$
- $e(6, 5)$

1 \{m(X, C) : c(C)\} 1 :- v(X).

:- e(X, Y), m(X, C), m(Y, C).

guess mapping $m$

verify $m(X) \neq m(Y)$
Example: Graph Colouring

Definition: graph colouring problem

Input: graph with vertices $V$ and edges $E \subseteq V \times V$, set of colors $C$. Is there a mapping $m : V \to C$ with $m(x) \neq m(y)$ for all $(x,y) \in E$?

```
c(r). c(g). c(b).
v(1). ... v(6).
e(1,2). e(1,3). e(1,4).
e(2,4). e(2,5). e(2,6).
e(3,1). e(3,4). e(3,5).
e(4,1). e(4,2).
e(5,3). e(5,4). e(5,6).
e(6,2). e(6,3). e(6,5).
```

```scheme
1 {m(X,C) : c(C)} 1 :- v(X).  
:- e(X,Y), m(X,C), m(Y,C).
```

Guess mapping $m$

Verify $m(X) \neq m(Y)$

Adopted from Potassco Slide Packages
Applications of ASP

- Automated product configuration
- Linux package manager
- Decision-support system for space shuttle
- Bioinformatics (diagnosis, inconsistency detection)
- General game playing

- Several implementations are available
- For this lecture: Clingo [www.potassco.org](http://www.potassco.org)
Overview of the Lecture

- Semantics of ASP programs
- Extensions of ASP programs
- Handling of variables in ASP
- ASP as modelling language
Prolog vs ASP

Consider the following logic program:

\[
\begin{align*}
a.
\quad c & \leftarrow a, b. \\
\quad d & \leftarrow a, \text{not } b.
\end{align*}
\]

\[
\begin{align*}
a.
\quad c & :\neg a, b. \\
\quad d & :\neg a, \text{not } b.
\end{align*}
\]
Prolog vs ASP

Consider the following logic program:

- $a$.
  - $c \leftarrow a, b$.
  - $d \leftarrow a, \neg b$.

- Prolog proves by SLD resolution:
Consider the following logic program:

- $a$.
  - $c \leftarrow a, b$.
  - $d \leftarrow a, \neg b$.

- Prolog proves by SLD resolution:
  - Proves $a$ (for $a$ is a fact)
Prolog vs ASP

Consider the following logic program:

- $a$.
  
  \[ c \leftarrow a, b. \]
  
  \[ d \leftarrow a, \text{not } b. \]

- Prolog proves by SLD resolution:
  
  - Proves $a$ (for $a$ is a fact)
  
  - Cannot prove $b$ (for $b$ is in no head)
Prolog vs ASP

Consider the following logic program:

- \( a \).
  
  \[ c \leftarrow a, b. \]
  
  \[ d \leftarrow a, \text{not } b. \]

- Prolog proves by SLD resolution:
  
  - Proves \( a \) (for \( a \) is a fact)
  
  - Cannot prove \( b \) (for \( b \) is in no head)
  
  - Cannot prove \( c \) (for cannot prove \( b \))
Prolog vs ASP

Consider the following logic program:

- \( a. \)
  - \( c \leftarrow a, b. \)
  - \( d \leftarrow a, \neg b. \)

- Prolog proves by SLD resolution:
  - Proves \( a \) (for \( a \) is a fact)
  - Cannot prove \( b \) (for \( b \) is in no head)
  - Cannot prove \( c \) (for cannot prove \( b \))
  - Proves \( d \) (for prove \( a \) but not \( b \))

Algorithm defines what Prolog does
Prolog vs ASP

Consider the following logic program:

- \( a \).
  - \( c \leftarrow a, b \).
  - \( d \leftarrow a, \text{not } b \).

Prolog proves by SLD resolution:

- Proves \( a \) (for \( a \) is a fact)
- Cannot prove \( b \) (for \( b \) is in no head)
- Cannot prove \( c \) (for cannot prove \( b \))
- Proves \( d \) (for prove \( a \) but not \( b \))

Algorithm defines what Prolog does

- What is the *semantics* of this logic program?
Consider the following logic program:

- $a.$
  - $c \leftarrow a, b.$
  - $d \leftarrow a, \neg b.$

Prolog proves by SLD resolution:

- Proves $a$ (for $a$ is a fact)
- Cannot prove $b$ (for $b$ is in no head)
- Cannot prove $c$ (for cannot prove $b$)
- Proves $d$ (for prove $a$ but not $b$)

Algorithm defines what Prolog does

What is the *semantics* of this logic program?

- Models:
  
  $M_1 = \begin{array}{cccc}
  a & b & c & d \\
  1 & 0 & 0 & 1 \\
  \end{array}$
  $M_2 = \begin{array}{cccc}
  a & b & c & d \\
  1 & 1 & 1 & 0 \\
  \end{array}$
  …
Prolog vs ASP

Consider the following logic program:

\[
\begin{align*}
  & a. \\
  & c \leftarrow a, b. \quad a \land b \rightarrow c \\
  & d \leftarrow a, \text{not } b. \quad a \land \neg b \rightarrow d
\end{align*}
\]

Prolog proves by SLD resolution:

- Proves \( a \) (for \( a \) is a fact)
- Cannot prove \( b \) (for \( b \) is in no head)
- Cannot prove \( c \) (for cannot prove \( b \))
- Proves \( d \) (for prove \( a \) but not \( b \))

Algorithm defines what Prolog does

What is the *semantics* of this logic program?

Models:

\[
\begin{align*}
  M_1 &= \begin{pmatrix}
    a & b & c & d \\
    1 & 0 & 0 & 1
  \end{pmatrix} \\
  M_2 &= \begin{pmatrix}
    a & b & c & d \\
    1 & 1 & 1 & 0
  \end{pmatrix}
\end{align*}
\]

\( M_1 \) corresponds to Prolog, what is special about \( M_1 \)?
Prolog vs ASP

Consider the following logic program:

- $a$
- $c \leftarrow a, b$.  \hspace{1cm}  a \land b \rightarrow c$
- $d \leftarrow a, \text{not } b$. \hspace{1cm}  a \land \neg b \rightarrow d$

Prolog proves by SLD resolution:

- Proves $a$ (for $a$ is a fact)
- Cannot prove $b$ (for $b$ is in no head)
- Cannot prove $c$ (for cannot prove $b$)
- Proves $d$ (for prove $a$ but not $b$)

Algorithm defines what Prolog does.

What is the *semantics* of this logic program?

- Models: $M_1 = \begin{bmatrix} a & b & c & d \\ 1 & 0 & 0 & 1 \end{bmatrix}$  \hspace{1cm}  $M_2 = \begin{bmatrix} a & b & c & d \\ 1 & 1 & 1 & 0 \end{bmatrix}$  \hspace{1cm}  ... 
- $M_1$ corresponds to Prolog, what is special about $M_1$?
- $M_1$ is a **stable model** a.k.a. **answer set**:
  - $M_1$ only satisfies justified propositions

ASP gives **semantics** to **logic programming**
The motivating guidelines behind stable model semantics are:

- A stable model satisfies all the rules of a logic program
- The reasoner shall not believe anything they are not forced to believe — the **rationality principle**
The motivating guidelines behind stable model semantics are:

- A stable model satisfies all the rules of a logic program
- The reasoner shall not believe anything they are not forced to believe — the *rationality principle*

Next: formalisation of this intuition

For now: only ground programs, i.e., no variables
### Definition: normal logic program (NLP)

A **normal logic program** $P$ is a set of (normal) rules of the form

$$A \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n.$$  

where $A, B_i, C_j$ are atomic propositions.

When $m = n = 0$, we omit the “$\leftarrow$” and just write $A$. 
Definition: normal logic program (NLP)

A **normal logic program** $P$ is a set of (normal) rules of the form

$$A \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n.$$ 

where $A, B_i, C_j$ are atomic propositions.

When $m = n = 0$, we omit the “$\leftarrow$” and just write $A$.

For such a rule $r$, we define:

- $\text{Head}(r) = \{A\}$
- $\text{Body}(r) = \{B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n\}$

In code, $r$ is written as $A : - B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n$. 

Syntax
**Semantics: Interpretation**

**Definition: interpretation, satisfaction**

An **interpretation** $S$ is a set of atomic propositions.

$S$ **satisfies** $A \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n$ iff $A \in S$ or some $B_i \notin S$ or some $C_j \in S$.

In English:

- $S$ satisfies rule iff $S$ satisfies the head or falsifies the body
- $S$ falsifies body iff $S$ falsifies some $B_i$ or satisfies some $C_j$
Definition: interpretation, satisfaction

An interpretation $S$ is a set of atomic propositions.

$S$ satisfies $A \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n$ iff $A \in S$ or some $B_i \notin S$ or some $C_j \in S$.

In English:

■ $S$ satisfies rule iff $S$ satisfies the head or falsifies the body

■ $S$ falsifies body iff $S$ falsifies some $B_i$ or satisfies some $C_j$

Ex.: Let $P = \{a. \ c \leftarrow a, b. \ d \leftarrow a, \not b.\}$
Semantics: Interpretation

Definition: interpretation, satisfaction

An **interpretation** $S$ is a set of atomic propositions.

$S$ **satisfies** $A \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n$ iff $A \in S$ or some $B_i \notin S$ or some $C_j \in S$.

In English:

- $S$ satisfies rule iff $S$ satisfies the head or falsifies the body
- $S$ falsifies body iff $S$ falsifies some $B_i$ or satisfies some $C_j$

**Ex.** Let $P = \{a.\ c \leftarrow a, b.\ d \leftarrow a, \text{not } b.\}$

$S = \{a, b, c\}$ satisfies $a$, but it does not satisfy (not $b$).
Semantics: Interpretation

Definition: interpretation, satisfaction

An **interpretation** $S$ is a set of atomic propositions.

$S$ satisfies $A \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n$ iff

$A \in S$ or some $B_i \notin S$ or some $C_j \in S$.

In English:

- S satisfies rule iff $S$ satisfies the head or falsifies the body
- S falsifies body iff $S$ falsifies some $B_i$ or satisfies some $C_j$

**Ex.** Let $P = \{a. \ c \leftarrow a, b. \ d \leftarrow a, \text{not } b.\}$

$S = \{a, b, c\}$ satisfies $a$, but it does not satisfy (not $b$).

It satisfies $c \leftarrow a, b$ because it satisfies the head because $c \in S$
Definition: interpretation, satisfaction

An interpretation $S$ is a set of atomic propositions.

$S$ satisfies $A \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n$ iff

$A \in S$ or some $B_i \notin S$ or some $C_j \in S$.

In English:

- $S$ satisfies rule iff $S$ satisfies the head or falsifies the body
- $S$ falsifies body iff $S$ falsifies some $B_i$ or satisfies some $C_j$

Ex.: Let $P = \{a. \ c \leftarrow a, b. \ d \leftarrow a, \text{not } b.\}$

$S = \{a, b, c\}$ satisfies $a$, but it does not satisfy $\text{not } b$.

It satisfies $c \leftarrow a, b$ because it satisfies the head because $c \in S$

It satisfies $d \leftarrow a, \text{not } b$ because it falsifies the body because $b \in S$
Definition: stable model for programs without negation

For \( P \) without negated literals:
\( S \) is a \textbf{stable model} of \( P \) iff

\( S \) is a minimal set (w.r.t. \( \subseteq \)) that satisfies all \( r \in P \).
**Definition: stable model for programs without negation**

For $P$ without negated literals:

$S$ is a **stable model** of $P$ iff

$S$ is a minimal set (w.r.t. $\subseteq$) that satisfies all $r \in P$.

**Ex.:** $P = \{ a. \ c \leftarrow a, b. \}$
Semantics without Negation

Definition: stable model for programs without negation

For $P$ without negated literals:
$S$ is a **stable model** of $P$ iff
$S$ is a minimal set (w.r.t. $\subseteq$) that satisfies all $r \in P$.

Ex.: $P = \{a. \ c \leftarrow a, b.\}$

$S_1 = \{a\}$ is a stable model of $P$
Semantics without Negation

Definition: stable model for programs without negation

For $P$ without negated literals:
$S$ is a **stable model** of $P$ iff
$S$ is a minimal set (w.r.t. $\subseteq$) that satisfies all $r \in P$.

Ex.: $P = \{a. \ c \leftarrow a, b.\}$
$S_1 = \{a\}$ is a stable model of $P$
$S_2 = \{a, b\}$ is not a stable model of $P$
Semantics without Negation

**Definition: stable model for programs without negation**

For $P$ without negated literals:

$S$ is a **stable model** of $P$ iff

$S$ is a minimal set (w.r.t. $\subseteq$) that satisfies all $r \in P$.

**Ex.:** $P = \{a. \ c \leftarrow a, b.\}$

$S_1 = \{a\}$ is a stable model of $P$

$S_2 = \{a, b\}$ is not a stable model of $P$

$S_3 = \{a, b, c\}$ is not a stable model of $P$
Semantics without Negation

**Definition: stable model for programs without negation**

For $P$ without negated literals:

$S$ is a **stable model** of $P$ iff

$S$ is a minimal set (w.r.t. $\subseteq$) that satisfies all $r \in P$.

**Example:**

$P = \{ a. c ← a, b. \}$

$S_1 = \{ a \}$ is a stable model of $P$

$S_2 = \{ a, b \}$ is not a stable model of $P$

$S_3 = \{ a, b, c \}$ is not a stable model of $P$

**Theorem: unique-model property**

If $P$ is negation-free (i.e., contains no $(\text{not } C)$), then there is exactly one stable model, which can be computed in linear time.
Computable stable model of a negation-free $P$ by *unit propagation*:

- $S^0 = \emptyset$
- $S^{i+1} = S^i \cup \bigcup_{r \in P: S \text{ satisfies } \text{Body}(r)} \text{Head}(r)$ until $S^{i+1} = S^i$
Semantics without Negation – Examples

Compute stable model of a negation-free $P$ by unit propagation:

- $S^0 = \{\}$
- $S^{i+1} = S^i \cup \bigcup_{r \in P: S \text{ satisfies Body}(r)} \text{Head}(r)$ until $S^{i+1} = S^i$

Ex.: $P_1 = \{a. \ b \leftarrow a.\}$

$S^0 = \{\} \quad S^1 = \{a\} \quad S^2 = \{a, b\}$ Fixpoint
Semantics without Negation – Examples

Compute stable model of a negation-free $P$ by unit propagation:

- $S^0 = \{\}$
- $S^{i+1} = S^i \cup \bigcup_{r \in P: S \text{ satisfies } \text{Body}(r)} \text{Head}(r)$ until $S^{i+1} = S^i$

Ex.: $P_1 = \{a. \ b \leftarrow a.\}$
$S^0 = \{\}$ $S^1 = \{a\}$ $S^2 = \{a, b\}$ Fixpoint

Ex.: $P_2 = \{a \leftarrow b. \ b \leftarrow a.\}$
$S^0 = \{\}$ Fixpoint
Semantics without Negation – Examples

Compute stable model of a negation-free $P$ by *unit propagation*:

- $S^0 = \{\}$
- $S^{i+1} = S^i \cup \bigcup_{r \in P: S \text{ satisfies Body}(r)} \text{Head}(r)$ until $S^{i+1} = S^i$

**Ex.:** $P_1 = \{a. \ b \leftarrow a.\}$

$S^0 = \{\} \quad S^1 = \{a\} \quad S^2 = \{a, b\} \quad \text{Fixpoint}$

**Ex.:** $P_2 = \{a \leftarrow b. \ b \leftarrow a.\}$

$S^0 = \{\} \quad \text{Fixpoint}$

**Ex.:** $P_3 = \{a \leftarrow b. \ b \leftarrow a. \ a.\}$

$S^0 = \{\} \quad S^1 = \{a\} \quad S^2 = \{a, b\} \quad \text{Fixpoint}$
Semantics with Negation

Definition: reduct

The **reduct** $P^S$ of $P$ relative to $S$ is the least set such that

if $A ← B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n ∈ P$ and $C_1, \ldots, C_n ∉ S$
then $A ← B_1, \ldots, B_m ∈ P^S$.

In English: for each rule $r$ from $P$,

- if $(\text{not } C) ∈ \text{Body}(r)$ for some $C ∈ S$: drop the rule
- else: remove all negated literals and add to $P^S$
**Definition: reduct**

The **reduct** $P^S$ of $P$ relative to $S$ is the least set such that

- **if** $A \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n \in P$ and $C_1, \ldots, C_n \notin S$
- **then** $A \leftarrow B_1, \ldots, B_m \in P^S$.

In English: for each rule $r$ from $P$,

- if $(\text{not } C) \in \text{Body}(r)$ for some $C \in S$: drop the rule
- else: remove all negated literals and add to $P^S$

**Ex.:** $P = \{a. \ c \leftarrow a, b. \ d \leftarrow a, \text{not } b.\}$
Definition: reduct

The **reduct** $P^S$ of $P$ relative to $S$ is the least set such that

- if $A ← B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n ∈ P$ and $C_1, \ldots, C_n ∉ S$
- then $A ← B_1, \ldots, B_m ∈ P^S$.

In English: for each rule $r$ from $P$,

- if $(\text{not } C) ∈ \text{Body}(r)$ for some $C ∈ S$: drop the rule
- else: remove all negated literals and add to $P^S$

**Ex.:** $P = \{a. \ c ← a, \ b. \ d ← a, \text{not } b.\}$

$S_1 = \{a\} \ \Rightarrow \ P^{S_1} = \{a. \ c ← a, \ b. \ d ← a, \text{not } b.\}$
Semantics with Negation

**Definition: reduct**

The **reduct** $P^S$ of $P$ relative to $S$ is the least set such that

if $A \leftarrow B_1, \ldots, B_m, \text{not} C_1, \ldots, \text{not} C_n \in P$ and $C_1, \ldots, C_n \notin S$

then $A \leftarrow B_1, \ldots, B_m \in P^S$.

In English: for each rule $r$ from $P$,
- if $(\text{not } C) \in \text{Body}(r)$ for some $C \in S$: drop the rule
- else: remove all negated literals and add to $P^S$

**Ex.:** $P = \{a. \ c \leftarrow a, b. \ d \leftarrow a, \text{not } b.\}$

$S_1 = \{a\} \Rightarrow P^{S_1} = \{a. \ c \leftarrow a, b. \ d \leftarrow a, \text{not } b.\}$

$S_2 = \{a, b\} \Rightarrow P^{S_2} = \{a. \ c \leftarrow a, b. \ d \leftarrow a, \text{not } b.\}$
Semantics with Negation

Definition: reduct

The **reduct** $P^S$ of $P$ relative to $S$ is the least set such that

if $A \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n \in P$ and $C_1, \ldots, C_n \notin S$
then $A \leftarrow B_1, \ldots, B_m \in P^S$.

In English: for each rule $r$ from $P$,

- if $(\text{not } C) \in \text{Body}(r)$ for some $C \in S$: drop the rule
- else: remove all negated literals and add to $P^S$

**Ex.:** $P = \{a. \ c \leftarrow a, b. \ d \leftarrow a, \text{not } b.\}$

$S_1 = \{a\} \Rightarrow P^{S_1} = \{a. \ c \leftarrow a, b. \ d \leftarrow a, \text{not } b.\}$

$S_2 = \{a, b\} \Rightarrow P^{S_2} = \{a. \ c \leftarrow a, b. \ d \leftarrow a, \text{not } b.\}$

$S_3 = \{a, d\} \Rightarrow P^{S_3} = \{a. \ c \leftarrow a, b. \ d \leftarrow a, \text{not } b.\}$
Semantics with Negation

**Definition: reduct**

The **reduct** $P^S$ of $P$ relative to $S$ is the least set such that

if $A \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n \in P$ and $C_1, \ldots, C_n \notin S$

then $A \leftarrow B_1, \ldots, B_m \in P^S$.

In English: for each rule $r$ from $P$,

- if $(\text{not } C) \in \text{Body}(r)$ for some $C \in S$: drop the rule
- else: remove all negated literals and add to $P^S$

**Ex.** $P = \{a. \ c \leftarrow a, b. \ d \leftarrow a, \text{not } b.\}$

$S_1 = \{a\} \Rightarrow P^{S_1} = \{a. \ c \leftarrow a, b. \ d \leftarrow a.\}$

$S_2 = \{a, b\} \Rightarrow P^{S_2} = \{a. \ c \leftarrow a, b.\}$

$S_3 = \{a, d\} \Rightarrow P^{S_3} = \{a. \ c \leftarrow a, b. \ d \leftarrow a.\}$
Semantics with Negation

**Definition: reduct**

The reduct $P^S$ of $P$ relative to $S$ is the least set such that:

- if $A ← B_1, \ldots, B_m, \neg C_1, \ldots, \neg C_n ∈ P$ and $C_1, \ldots, C_n ∉ S$
- then $A ← B_1, \ldots, B_m ∈ P^S$.

In English: for each rule $r$ from $P$,

- if $(\neg C) ∈ \text{Body}(r)$ for some $C ∈ S$: drop the rule
- else: remove all negated literals and add to $P^S$

**Ex.:** $P = \{a. c ← a, b. d ← a, \neg b.\}$

- $S_1 = \{a\} \Rightarrow P^{S_1} = \{a. c ← a, b. d ← a.\}$
- $S_2 = \{a, b\} \Rightarrow P^{S_2} = \{a. c ← a, b.\}$
- $S_3 = \{a, d\} \Rightarrow P^{S_3} = \{a. c ← a, b. d ← a.\}$

**Definition: stable model for programs with negation**

For $P$ with negated literals:

$S$ is a **stable model** of $P$ iff $S$ is a stable model of $P^S$. 
Semantics with Negation

Definition: reduct

The reduct $P^S$ of $P$ relative to $S$ is the least set such that
if $A \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n \in P$ and $C_1, \ldots, C_n \notin S$
then $A \leftarrow B_1, \ldots, B_m \in P^S$.

In English: for each rule $r$ from $P$,
- if $(\text{not } C) \in \text{Body}(r)$ for some $C \in S$: drop the rule
- else: remove all negated literals and add to $P^S$

Ex.: $P = \{a. \ c \leftarrow a, b. \ d \leftarrow a, \text{not } b.\}$
$S_1 = \{a\} \Rightarrow P^{S_1} = \{a. \ c \leftarrow a, b. \ d \leftarrow a.\}$  
$S_2 = \{a, b\} \Rightarrow P^{S_2} = \{a. \ c \leftarrow a, b.\}$
$S_3 = \{a, d\} \Rightarrow P^{S_3} = \{a. \ c \leftarrow a, b. \ d \leftarrow a.\}$

Definition: stable model for programs with negation

For $P$ with negated literals:
$S$ is a stable model of $P$ iff $S$ is a stable model of $P^S$. 
**Semantics with Negation**

**Definition: reduct**

The **reduct** $P^S$ of $P$ relative to $S$ is the least set such that

- if $A \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n \in P$ and $C_1, \ldots, C_n \notin S$
- then $A \leftarrow B_1, \ldots, B_m \in P^S$.

In English: for each rule $r$ from $P$,

- if $(\text{not } C) \in \text{Body}(r)$ for some $C \in S$: drop the rule
- else: remove all negated literals and add to $P^S$

**Ex.:** $P = \{a. \ c \leftarrow a, b. \ d \leftarrow a, \text{not } b.\}$

- $S_1 = \{a\} \Rightarrow P^{S_1} = \{a. \ c \leftarrow a, b. \ d \leftarrow a.\}$
- $S_2 = \{a, b\} \Rightarrow P^{S_2} = \{a. \ c \leftarrow a, b.\}$
- $S_3 = \{a, d\} \Rightarrow P^{S_3} = \{a. \ c \leftarrow a, b. \ d \leftarrow a.\}$

**Definition: stable model for programs with negation**

For $P$ with negated literals:

- $S$ is a **stable model** of $P$ iff $S$ is a stable model of $P^S$. 
Semantics with Negation

**Definition: reduct**

The **reduct** $P^S$ of $P$ relative to $S$ is the least set such that

- if $A \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n \in P$ and $C_1, \ldots, C_n \notin S$
- then $A \leftarrow B_1, \ldots, B_m \in P^S$.

In English: for each rule $r$ from $P$,
- if $(\text{not } C) \in \text{Body}(r)$ for some $C \in S$: drop the rule
- else: remove all negated literals and add to $P^S$

**Ex.:** $P = \{a. \ c \leftarrow a, b. \ d \leftarrow a, \text{not } b.\}$

$S_1 = \{a\} \Rightarrow P^{S_1} = \{a. \ c \leftarrow a, b. \ d \leftarrow a.\}$ *X*

$S_2 = \{a, b\} \Rightarrow P^{S_2} = \{a. \ c \leftarrow a, b.\}$ *X*

$S_3 = \{a, d\} \Rightarrow P^{S_3} = \{a. \ c \leftarrow a, b. \ d \leftarrow a.\}$ *

**Definition: stable model for programs with negation**

For $P$ with negated literals:

$S$ is a **stable model** of $P$ iff $S$ is a stable model of $P^S$. 
Semantics with Negation – Examples

Ex.: \( P = \{ a \leftarrow \neg b. \quad b \leftarrow \neg a. \} \)
Semantics with Negation – Examples

Ex.: $P = \{ a \leftarrow \text{not } b. \ b \leftarrow \text{not } a. \}$

$S_1 = \{\} \quad \Rightarrow \quad P^{S_1} = \{\}$
Semantics with Negation – Examples

\[
\text{Ex.: } P = \{ a \leftarrow \text{not } b. \quad b \leftarrow \text{not } a. \} \\
S_1 = \{ \} \quad \Rightarrow \quad P^{S_1} = \{ a \leftarrow \text{not } b. \quad b \leftarrow \text{not } a. \} 
\]
Semantics with Negation – Examples

\textbf{Ex.}: \( P = \{a \leftarrow \text{not } b. \ b \leftarrow \text{not } a.\} \)

\( S_1 = \{\} \quad \Rightarrow \quad P^{S_1} = \{a \leftarrow \text{not } b. \ b \leftarrow \text{not } a.\} \)

\( S_2 = \{a\} \quad \Rightarrow \quad P^{S_2} = \)
Semantics with Negation – Examples

Ex.: $P = \{ a \leftarrow \text{not } b, \ b \leftarrow \text{not } a. \}$

$S_1 = \{ \} \quad \Rightarrow \quad P^{S_1} = \{ a \leftarrow \text{not } b, \ b \leftarrow \text{not } a. \}$

$S_2 = \{ a \} \quad \Rightarrow \quad P^{S_2} = \{ a \leftarrow \text{not } b, \ b \leftarrow \text{not } a. \}$
Ex.: $P = \{a \leftarrow \neg b. \quad b \leftarrow \neg a.\}$

$S_1 = \{\} \quad \Rightarrow \quad P^{S_1} = \{a \leftarrow \neg b. \quad b \leftarrow \neg a.\}$

$S_2 = \{a\} \quad \Rightarrow \quad P^{S_2} = \{a \leftarrow \neg b. \quad b \leftarrow \neg a.\}$

$S_3 = \{b\} \quad \Rightarrow \quad P^{S_3} = \{a \leftarrow \neg b. \quad b \leftarrow \neg a.\}$
**Semantics with Negation – Examples**

**Ex.:** $P = \{a \leftarrow \text{not } b. \ b \leftarrow \text{not } a.\}$

$S_1 = \{\} \quad \Rightarrow \quad P^{S_1} = \{a \leftarrow \text{not } b. \ b \leftarrow \text{not } a.\}$

$S_2 = \{a\} \quad \Rightarrow \quad P^{S_2} = \{a \leftarrow \text{not } b. \ b \leftarrow \text{not } a.\}$

$S_3 = \{b\} \quad \Rightarrow \quad P^{S_3} = \{a \leftarrow \text{not } b. \ b \leftarrow \text{not } a.\}$

$S_4 = \{a, b\} \quad \Rightarrow \quad P^{S_4} = \{a \leftarrow \text{not } b. \ b \leftarrow \text{not } a.\}$
Ex.: $P = \{ a \leftarrow \text{not } b. \quad b \leftarrow \text{not } a. \}$

$S_1 = \{\} \quad \Rightarrow \quad PS_1 = \{ a \leftarrow \text{not } b. \quad b \leftarrow \text{not } a. \}$

$S_2 = \{a\} \quad \Rightarrow \quad PS_2 = \{ a \leftarrow \text{not } b. \quad b \leftarrow \text{not } a. \}$

$S_3 = \{b\} \quad \Rightarrow \quad PS_3 = \{ a \leftarrow \text{not } b. \quad b \leftarrow \text{not } a. \}$

$S_4 = \{a, b\} \quad \Rightarrow \quad PS_4 = \{ a \leftarrow \text{not } b. \quad b \leftarrow \text{not } a. \}$
Semantics with Negation – Examples

Ex.: \( P = \{ a \leftarrow \text{not } b. \ b \leftarrow \text{not } a. \} \)

\( S_1 = \{\} \Rightarrow P^{S_1} = \{ a \leftarrow \text{not } b. \ b \leftarrow \text{not } a. \} \)

\( S_2 = \{ a \} \Rightarrow P^{S_2} = \{ a \leftarrow \text{not } b. \ b \leftarrow \text{not } a. \} \)

\( S_3 = \{ b \} \Rightarrow P^{S_3} = \{ a \leftarrow \text{not } b. \ b \leftarrow \text{not } a. \} \)

\( S_4 = \{ a, b \} \Rightarrow P^{S_4} = \{ a \leftarrow \text{not } b. \ b \leftarrow \text{not } a. \} \)
Semantics with Negation – Examples

Ex.: $P = \{ a \leftarrow \neg b. \quad b \leftarrow \neg a. \}$

$S_1 = \{\} \quad \Rightarrow \quad P^{S_1} = \{ a. \quad b \}$

$S_2 = \{a\} \quad \Rightarrow \quad P^{S_2} = \{a.\}$

$S_3 = \{b\} \quad \Rightarrow \quad P^{S_3} = \{b.\}$

$S_4 = \{a, b\} \quad \Rightarrow \quad P^{S_4} = \{\}$
**Semantics with Negation – Examples**

Ex.: \( P = \{a \leftarrow \text{not } b. \quad b \leftarrow \text{not } a.\} \)

\[ S_1 = \{\} \quad \Rightarrow \quad P^{S_1} = \{a. \quad b\} \)

\[ S_2 = \{a\} \quad \Rightarrow \quad P^{S_2} = \{a.\} \)

\[ S_3 = \{b\} \quad \Rightarrow \quad P^{S_3} = \{b.\} \)

\[ S_4 = \{a, b\} \quad \Rightarrow \quad P^{S_4} = \{\} \)

\( \times \)
\textbf{Semantics with Negation – Examples}

\begin{align*}
\text{Ex.}: \quad P = \{a \leftarrow \text{not } b. \quad b \leftarrow \text{not } a.\} \\
S_1 = \{\} \quad \Rightarrow \quad P^{S_1} = \{a. \quad b\} \quad \checkmark \\
S_2 = \{a\} \quad \Rightarrow \quad P^{S_2} = \{a.\} \quad \times \\
S_3 = \{b\} \quad \Rightarrow \quad P^{S_3} = \{b.\} \\
S_4 = \{a, b\} \quad \Rightarrow \quad P^{S_4} = \{\} \\
\end{align*}
Semantics with Negation – Examples

**Example:** $P = \{ a \leftarrow \text{not } b, \quad b \leftarrow \text{not } a. \}$

- $S_1 = \{\} \quad \Rightarrow \quad P^{S_1} = \{a, b\}$  
- $S_2 = \{a\} \quad \Rightarrow \quad P^{S_2} = \{a\}$  
- $S_3 = \{b\} \quad \Rightarrow \quad P^{S_3} = \{b\}$  
- $S_4 = \{a, b\} \quad \Rightarrow \quad P^{S_4} = \{\}$
Semantics with Negation – Examples

\[ \text{Ex.: } P = \{ a \leftarrow \text{not } b. \quad b \leftarrow \text{not } a. \} \]

\[ S_1 = \{ \} \quad \Rightarrow \quad P^{S_1} = \{ a. \quad b \} \quad \checkmark \]

\[ S_2 = \{ a \} \quad \Rightarrow \quad P^{S_2} = \{ a. \} \quad \checkmark \]

\[ S_3 = \{ b \} \quad \Rightarrow \quad P^{S_3} = \{ b. \} \quad \checkmark \]

\[ S_4 = \{ a, b \} \quad \Rightarrow \quad P^{S_4} = \{ \} \quad \times \]

Two stable models!
Semantics with Negation – Examples

\[ P = \{ a \leftarrow \neg b, \ b \leftarrow \neg a \} \]

- \( S_1 = \{ \} \Rightarrow P^{S_1} = \{ a, b \} \) ☒
- \( S_2 = \{ a \} \Rightarrow P^{S_2} = \{ a \} \) ☒
- \( S_3 = \{ b \} \Rightarrow P^{S_3} = \{ b \} \) ☒
- \( S_4 = \{ a, b \} \Rightarrow P^{S_4} = \{ \} \) ☒

Two stable models!

\[ P = \{ a \leftarrow \neg a \} \]

\[ P = \{ a \leftarrow \neg b, \ b \leftarrow \neg a \} \]

- \( S_1 = \{ \} \Rightarrow P^{S_1} = \{ a, b \} \) ☒
- \( S_2 = \{ a \} \Rightarrow P^{S_2} = \{ a \} \) ☒
- \( S_3 = \{ b \} \Rightarrow P^{S_3} = \{ b \} \) ☒
- \( S_4 = \{ a, b \} \Rightarrow P^{S_4} = \{ \} \) ☒

Two stable models!
Ex.: $P = \{a \leftarrow \text{not } b. \quad b \leftarrow \text{not } a.\}$

$S_1 = \{\} \quad \Rightarrow \quad P^{S_1} = \{a. \quad b\}$

$S_2 = \{a\} \quad \Rightarrow \quad P^{S_2} = \{a.\}$

$S_3 = \{b\} \quad \Rightarrow \quad P^{S_3} = \{b.\}$

$S_4 = \{a, b\} \quad \Rightarrow \quad P^{S_4} = \{}$

Two stable models!

Ex.: $P = \{a \leftarrow \text{not } a.\}$

$S_1 = \{\} \quad \Rightarrow \quad P^{S_1} =$
Semantics with Negation – Examples

Ex.: \( P = \{ a \leftarrow \text{not} \ b. \ b \leftarrow \text{not} \ a. \} \)

\[ S_1 = \{ \} \quad \Rightarrow \quad P^{S_1} = \{ a, b \} \quad \checkmark \]

\[ S_2 = \{ a \} \quad \Rightarrow \quad P^{S_2} = \{ a \} \quad \checkmark \]

\[ S_3 = \{ b \} \quad \Rightarrow \quad P^{S_3} = \{ b \} \quad \checkmark \]

\[ S_4 = \{ a, b \} \quad \Rightarrow \quad P^{S_4} = \{ \} \quad \times \]

Two stable models!

Ex.: \( P = \{ a \leftarrow \text{not} \ a. \} \)

\[ S_1 = \{ \} \quad \Rightarrow \quad P^{S_1} = \{ a \leftarrow \text{not} \ a. \} \]
Semantics with Negation – Examples

Ex.: \( P = \{ a \leftarrow \text{not } b. \quad b \leftarrow \text{not } a. \} \)

\[
\begin{align*}
S_1 &= \{\} & \Rightarrow \quad P^{S_1} &= \{a. \quad b\} \\
S_2 &= \{a\} & \Rightarrow \quad P^{S_2} &= \{a.\} \\
S_3 &= \{b\} & \Rightarrow \quad P^{S_3} &= \{b.\} \\
S_4 &= \{a, b\} & \Rightarrow \quad P^{S_4} &= \{\} \\
\end{align*}
\]

Two stable models!

Ex.: \( P = \{ a \leftarrow \text{not } a. \} \)

\[
\begin{align*}
S_1 &= \{\} & \Rightarrow \quad P^{S_1} &= \{a \leftarrow \text{not } a.\} \\
S_2 &= \{a\} & \Rightarrow \quad P^{S_2} &= \\
\end{align*}
\]
Semantics with Negation – Examples

Ex.: $P = \{ a \leftarrow \text{not } b. \quad b \leftarrow \text{not } a. \}$

$S_1 = \{\} \quad \Rightarrow \quad P^S_1 = \{ a. \quad b \}$
$S_2 = \{a\} \quad \Rightarrow \quad P^S_2 = \{ a. \}$
$S_3 = \{b\} \quad \Rightarrow \quad P^S_3 = \{ b. \}$
$S_4 = \{a, b\} \quad \Rightarrow \quad P^S_4 = \{\}$

Two stable models!

Ex.: $P = \{ a \leftarrow \text{not } a. \}$

$S_1 = \{\} \quad \Rightarrow \quad P^S_1 = \{ a \leftarrow \text{not } a. \}$
$S_2 = \{a\} \quad \Rightarrow \quad P^S_2 = \{ a \leftarrow \text{not } a. \}$
Semantics with Negation – Examples

\textbf{Ex.}: \[ P = \{ a \leftarrow \text{not } b. \quad b \leftarrow \text{not } a. \} \]

\[ S_1 = \{ \} \quad \Rightarrow \quad P^{S_1} = \{ a. \quad b \} \quad \times \]
\[ S_2 = \{ a \} \quad \Rightarrow \quad P^{S_2} = \{ a. \} \quad \checkmark \]
\[ S_3 = \{ b \} \quad \Rightarrow \quad P^{S_3} = \{ b. \} \quad \checkmark \]
\[ S_4 = \{ a, b \} \quad \Rightarrow \quad P^{S_4} = \{ \} \quad \times \]

Two stable models!

\textbf{Ex.}: \[ P = \{ a \leftarrow \text{not } a. \} \]

\[ S_1 = \{ \} \quad \Rightarrow \quad P^{S_1} = \{ a. \} \]
\[ S_2 = \{ a \} \quad \Rightarrow \quad P^{S_2} = \{ \} \]
Semantics with Negation – Examples

Ex.: \( P = \{ a \leftarrow \text{not } b. \quad b \leftarrow \text{not } a. \} \)

\[
\begin{align*}
S_1 &= \{\} \quad \Rightarrow \quad P^{S_1} = \{ a. \quad b \} \\
S_2 &= \{a\} \quad \Rightarrow \quad P^{S_2} = \{ a. \} \\
S_3 &= \{b\} \quad \Rightarrow \quad P^{S_3} = \{ b. \} \\
S_4 &= \{a, b\} \quad \Rightarrow \quad P^{S_4} = \{}
\end{align*}
\]

Two stable models!

Ex.: \( P = \{ a \leftarrow \text{not } a. \} \)

\[
\begin{align*}
S_1 &= \{\} \quad \Rightarrow \quad P^{S_1} = \{ a. \} \\
S_2 &= \{a\} \quad \Rightarrow \quad P^{S_2} = \{}
\end{align*}
\]
Semantics with Negation – Examples

Ex.: $P = \{ a \leftarrow \text{not } b, \ b \leftarrow \text{not } a. \}$

$S_1 = \{ \} \quad \Rightarrow \quad P^{S_1} = \{ a, \ b \}$

$S_2 = \{ a \} \quad \Rightarrow \quad P^{S_2} = \{ a. \}$

$S_3 = \{ b \} \quad \Rightarrow \quad P^{S_3} = \{ b. \}$

$S_4 = \{ a, b \} \quad \Rightarrow \quad P^{S_4} = \{ \}$

Two stable models!

Ex.: $P = \{ a \leftarrow \text{not } a. \}$

$S_1 = \{ \} \quad \Rightarrow \quad P^{S_1} = \{ a. \}$

$S_2 = \{ a \} \quad \Rightarrow \quad P^{S_2} = \{ \}$

No stable model!
Definition: reduct

The **reduct** $P^S$ of $P$ relative to $S$ is the least set such that

\[
\text{if } A \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n \in P \text{ and } C_1, \ldots, C_n \notin S \text{ then } A \leftarrow B_1, \ldots, B_m \in P^S.
\]

Definition: stable model

If $P$ contains no (not $C$):

$S$ is a **stable model** of $P$ iff

$S$ is a minimal set (w.r.t. $\subseteq$) that satisfies all $r \in P$.

If $P$ contains (not $C$):

$S$ is a **stable model** of $P$ iff $S$ is a stable model of $P^S$.

Theorem: necessary satisfaction condition

If $S$ is a stable model and $A \in S$,
then $S$ satisfies some $r \in P$ with $A \in \text{Head}(r)$.
Semantics – Examples

Ex.: \( P = \{ a \leftarrow a. \quad b \leftarrow \text{not} \ a. \} \)

\[ S \quad \quad \quad p^S \]

Stable model?

Ex.: \( P = \{ a \leftarrow \text{not} \ b. \quad b \leftarrow \text{not} \ c. \} \)

\[ S \quad \quad \quad p^S \]

Stable model?

Example on paper
Overview of the Lecture

- Semantics of ASP programs
- Extensions of ASP programs
- Handling of variables in ASP
- ASP as modelling language
Definition: integrity constraint

An **integrity constraint** is a rule $r$ of the form

$$\leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n$$

$S$ satisfies $r$ iff some $B_i \notin S$ or some $C_j \in S$.

$P^S$ contains $\leftarrow B_1, \ldots, B_m$ iff $P$ contains $r$ and $C_1, \ldots, C_n \notin S$. 
Integrity Constraints

Definition: integrity constraint

An **integrity constraint** is a rule \( r \) of the form

\[
\begin{align*}
&\leftarrow B_1, \ldots, B_m, \neg C_1, \ldots, \neg C_n \\
\end{align*}
\]

\( S \) satisfies \( r \) iff some \( B_i \not\in S \) or some \( C_j \in S \).

\( P^S \) contains \( \leftarrow B_1, \ldots, B_m \) iff \( P \) contains \( r \) and \( C_1, \ldots, C_n \not\in S \).

Theorem: reduction to normal rules

Let \( P' \) be like \( P \) except that every integrity constraint

\[
\begin{align*}
&\leftarrow B_1, \ldots, B_m, \neg C_1, \ldots, \neg C_n \\
\end{align*}
\]

is replaced with

\[
\begin{align*}
&\text{dummy} \leftarrow B_1, \ldots, B_m, \neg C_1, \ldots, \neg C_n, \neg \text{dummy} \\
\end{align*}
\]

for some new atom \( \text{dummy} \).

Then \( P \) and \( P' \) have the same stable models.
Choice Rules

**Definition: choice rule**

A *choice rule* is a rule the form

\[
\{A_1, \ldots, A_k\} \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n
\]

which allows any subset of \(\{A_1, \ldots, A_k\}\) in a stable model.
Choice Rules

Definition: choice rule

A **choice rule** is a rule the form
\[
\{A_1, \ldots, A_k\} \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n
\]
which allows any subset of \(\{A_1, \ldots, A_k\}\) in a stable model.

Theorem: reduction to normal rules

A choice rule can be encoded by \(2k + 1\) normal rules using \(2k + 1\) new atoms.
Choice Rules

Definition: choice rule

A choice rule is a rule the form

\[ \{A_1, \ldots, A_k\} \leftarrow B_1, \ldots, B_m, \text{not } C_1, \ldots, \text{not } C_n \]

which allows any subset of \( \{A_1, \ldots, A_k\} \) in a stable model.

Theorem: reduction to normal rules

A choice rule can be encoded by \(2k + 1\) normal rules using \(2k + 1\) new atoms.

Further extensions:

- **Conditional literals**: \(\{A : B\}\)
  
  Ex.: \(\{m(v, C) : c(C)\}\) expands to \(\{m(v, r), m(v, g), m(v, b)\}\)

- **Cardinality constraints**: \(\text{min } \{A_1, \ldots, A_k\}\) \(\text{max}\)
  
  Ex.: \(1\) \(\{m(v, r), m(v, g), m(v, b)\}\) \(1\)
Negation in the Rule Head

Definition: rules with negated head

A rule with **negated head** is of the form

\[ \text{not} \ A \leftarrow B_1, \ldots, B_m, \text{not} \ C_1, \ldots, \text{not} \ C_n \]
Negation in the Rule Head

**Definition: rules with negated head**

A rule with **negated head** is of the form

\[
\text{not} A \leftarrow B_1, \ldots, B_m, \text{not} C_1, \ldots, \text{not} C_n
\]

**Theorem: reduction to normal rules**

Let \( P' \) be like \( P \) except that every rule with negated head

\[
\text{not} A \leftarrow B_1, \ldots, B_m, \text{not} C_1, \ldots, \text{not} C_n
\]

is replaced with

\[
\leftarrow B_1, \ldots, B_m, \text{not} C_1, \ldots, \text{not} C_n, \text{not} \text{dummy}
\]

and

\[
dummy \leftarrow \text{not} A
\]

for some new atom \( dummy \).

Then \( P \) and \( P' \) have the same stable models (modulo \( dummy \) propositions).
Theorem: complexity of NLPs without negations

Is $S$ a stable model of a negation-free $P$? – **Linear time**
Does a negation-free $P$ have a stable model? – **Constant** (yes, one)

Theorem: complexity of NLPs with negations

Is $S$ a stable model of $P$? – **Linear time**
Does $P$ have a stable model? – **NP-complete**

**Note:** integrity constraints, choice rules, negation in heads **preserve complexity** (program grows only polynomially)
Overview of the Lecture

- Semantics of ASP programs
- Extensions of ASP programs
- Handling of variables in ASP
- ASP as modelling language
Atomic propositions may now contain variables, e.g.,
\[ p(X, Z) \leftarrow e(X, Y), p(Y, Z). \]
Atomic propositions may now contain variables, e.g.,
\[ p(X, Z) \leftarrow e(X, Y), p(Y, Z). \]

Herbrand universe
- \( U \) contains all constants from \( P \)
- \( U \) contains all \( f(t_1, \ldots, t_k) \) from \( P \) if \( f \) is a \( k \)-ary function in \( P \) and \( U \) contains \( t_1, \ldots, t_k \)
Programs with Variables

- Atomic propositions may now contain variables, e.g.,
  \[ p(X, Z) \leftarrow e(X, Y), p(Y, Z). \]

- Herbrand universe
  - \( U \) contains all constants from \( P \)
  - \( U \) contains all \( f(t_1, \ldots, t_k) \) from \( P \) if \( f \) is a \( k \)-ary function in \( P \) and \( U \) contains \( t_1, \ldots, t_k \)

- ASP **grounds** variables with Herbrand universe
  - Unlike Prolog: instantiation instead of unification
  - Caution: the ground program may grow exponentially
  - Caution: function symbols make grounding Turing-complete
  - If \( P \) is finite and mentions only constants, grounding is finite
Programs with Variables

- $f(X) \leftarrow b(X), \text{not } a(X)$.
  - $a(X) \leftarrow p(X)$.
  - $b($sam$)$.
  - $b($tweety$)$.
  - $p($tweety$)$.

- $f($sam$) \leftarrow b($sam$), \text{not } a($sam$)$.
  - $f($tweety$) \leftarrow b($tweety$), \text{not } a($tweety$)$.
  - $a($sam$) \leftarrow p($sam$)$.
  - $a($tweety$) \leftarrow p($tweety$)$.
  - $b($sam$)$.
  - $b($tweety$)$.
  - $p($tweety$)$.
Overview of the Lecture

- Semantics of ASP programs
- Extensions of ASP programs
- Handling of variables in ASP
- ASP as modelling language
Typical ASP structure:

- Problem **instance**: a set of facts
- Problem **class**: a set of rules
  - Generator rules: often choice rules
  - Test rules: often integrity constraints

Ideal modeling is **uniform**: problem class encoding fits all instances

Semantically equivalent encodings may differ immensely in performance!
Example: Non-monotonic Reasoning

Tweety the penguin:
- (Normal) Birds fly.
- Penguins are abnormal.
- Tweety is a bird. So Tweety flies.
- Tweety is a penguin. So Tweety doesn’t fly.
Example: Non-monotonic Reasoning

Tweety the penguin:
- (Normal) Birds fly.
- Penguins are abnormal.
- Tweety is a bird. So Tweety flies.
- Tweety is a penguin. So Tweety doesn’t fly.

\[
U = \{ f(X) \leftarrow b(X), \text{not } a(X). \quad a(X) \leftarrow p(X). \quad b(t). \} \\
P = \{ f(t) \leftarrow b(t), \text{not } a(t). \quad a(t) \leftarrow p(t). \quad b(t). \}
\]
Example: Non-monotonic Reasoning

Tweety the penguin:
- (Normal) Birds fly.
- Penguins are abnormal.
- Tweety is a bird. So Tweety flies.
- Tweety is a penguin. So Tweety doesn’t fly.

\[ U = \{ f(X) \leftarrow b(X), \text{not } a(X). \quad a(X) \leftarrow p(X). \quad b(t). \} \]
\[ P = \{ f(t) \leftarrow b(t), \text{not } a(t). \quad a(t) \leftarrow p(t). \quad b(t). \} \]

\[ S_1 = \{ b(t), f(t) \} \quad \Rightarrow \quad P^S_1 = \{ f(t) \leftarrow b(t), \text{not } a(t). \quad a(t) \leftarrow p(t). \quad b(t) \}. \checkmark \]
\[ S_2 = \{ a(t), b(t), p(t) \} \quad \Rightarrow \quad P^S_2 = \{ f(t) \leftarrow b(t), \text{not } a(t). \quad a(t) \leftarrow p(t). \quad b(t) \}. \times \]

Tweety flies!
Example: Non-monotonic Reasoning

Tweety the penguin:
- (Normal) Birds fly.
- Penguins are abnormal.
- Tweety is a bird. So Tweety flies.
- Tweety is a penguin. So Tweety doesn’t fly.

\[ U = \{ f(X) \leftarrow b(X), \text{not } a(X). \ a(X) \leftarrow p(X). \ b(t). \} \]
\[ P = \{ f(t) \leftarrow b(t), \text{not } a(t). \ a(t) \leftarrow p(t). \ b(t). \} \]

\[ S_1 = \{ b(t), f(t) \} \quad \Rightarrow \quad P^{S_1} = \{ f(t) \leftarrow b(t), \text{not } a(t). \ a(t) \leftarrow p(t). \ b(t). \} \quad \checkmark \]
\[ S_2 = \{ a(t), b(t), p(t) \} \quad \Rightarrow \quad P^{S_2} = \{ f(t) \leftarrow b(t), \text{not } a(t). \ a(t) \leftarrow p(t). \ b(t). \} \quad \xmark \]

Tweety flies!

\[ S_1 = \{ b(t), f(t) \} \quad \Rightarrow \quad (P \cup \{ p(t). \})^{S_1} = P_{2}^{S_1} \cup \{ p(t). \} \quad \xmark \]
\[ S_2 = \{ a(t), b(t), p(t) \} \quad \Rightarrow \quad (P \cup \{ p(t). \})^{S_2} = P_{2}^{S_1} \cup \{ p(t). \} \quad \checkmark \]

Tweety doesn’t fly.
Definition: Hamilton cycle problem

Input: graph with vertex set $V$ and edges $E \subseteq V \times V$. Is there a cycle that visits every vertex exactly once?
Example: Hamilton Cycle

Definition: Hamilton cycle problem
Input: graph with vertex set \( V \) and edges \( E \subseteq V \times V \).
Is there a cycle that visits every vertex exactly once?

\[
\begin{align*}
\{p(X, Y)\} & \leftarrow e(X, Y). \\
r(X) & \leftarrow p(1, X). \\
r(Y) & \leftarrow r(X), p(X, Y). \\
& \leftarrow 2 \{p(X, Y)\}, v(X). \\
& \leftarrow 2 \{p(X, Y)\}, v(Y). \\
& \leftarrow \text{not } r(X), v(X).
\end{align*}
\]
Example: $N$-Queens

**Definition: $N$-queens problem**

Place $N$ queens on a $N \times N$ chessboard so that they do not attack each other, i.e., share no row, column, or diagonal.

Program on paper
Example: $N$-Queens

Definition: $N$-queens problem

Place $N$ queens on a $N \times N$ chessboard so that they do not attack each other, i.e., share no row, column, or diagonal.

Program on paper