Answer Set Programming

(1) Semantics of ASP programs

Abdallah Saffidine

COMP4418

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:

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

```
a.
c :- a, b.
d :- a, not b.
```

Prolog vs ASP

Consider the following logic program:

- $a.$
  $c \leftarrow a, b.$
  $d \leftarrow a, \text{not}\, b.$
- Prolog proves by SLD resolution:

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)

Prolog vs ASP

Consider the following logic program:

- $a.$
  $c \leftarrow a, b.$
  $d \leftarrow a, \mathrm{not}\, b.$
- 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:

- $a.$
  $c \leftarrow a, b.$
  $d \leftarrow a, \mathrm{not}\, b.$
- 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:

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

Prolog vs ASP

Consider the following logic program:

- $a.$
  $c \leftarrow a, b.$
  $d \leftarrow a, \mathrm{not}\, b.$
- 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?

Prolog vs ASP

Consider the following logic program:

- $a.$ $\quad\quad a$
  $c \leftarrow a, b.$ $\quad\quad a \wedge b \rightarrow c$
  $d \leftarrow a, \text{not}\, b.$ $\quad\quad a \wedge \neg b \rightarrow d$
- 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: $M_1 = \begin{array}{|c|c|c|c|}\hline a & b & c & d \\ \hline 1 & 0 & 0 & 1 \\ \hline\end{array}$ $\quad M_2 = \begin{array}{|c|c|c|c|}\hline a & b & c & d \\ \hline 1 & 1 & 1 & 0 \\ \hline\end{array}$ $\quad \dots$

Prolog vs ASP

Consider the following logic program:

- $a.$ $\quad\quad\quad\quad\quad\quad\quad\quad$ a

  $c \leftarrow a, b.$ $\quad\quad\quad\quad$ $a \wedge b \rightarrow c$

  $d \leftarrow a, \text{not}\, b.$ $\quad\quad$ $a \wedge \neg b \rightarrow d$
- 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: $M_1 = \begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline 1 & 0 & 0 & 1 \\ \hline \end{array}$ $\quad M_2 = \begin{array}{|c|c|c|c|} \hline a & b & c & d \\ \hline 1 & 1 & 1 & 0 \\ \hline \end{array}$ $\quad\ldots$
  - M_1 corresponds to Prolog, what is special about M_1?

Prolog vs ASP

Consider the following logic program:

- $a.$ $\qquad\qquad\qquad\qquad a$
  $c \leftarrow a, b.$ $\qquad\qquad a \wedge b \rightarrow c$
  $d \leftarrow a, \text{not}\, b.$ $\qquad\quad a \wedge \neg b \rightarrow d$
- 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: $M_1 = \begin{array}{|c|c|c|c|}\hline a & b & c & d \\ \hline 1 & 0 & 0 & 1 \\ \hline\end{array}$ $\quad M_2 = \begin{array}{|c|c|c|c|}\hline a & b & c & d \\ \hline 1 & 1 & 1 & 0 \\ \hline\end{array}$ $\quad \ldots$
  - 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**

Intuition

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**

Intuition

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

Syntax

Definition: normal logic program (NLP)

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

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

where A, B_i, C_j are atomic propositions.
When $m = n = 0$, we omit the "$\leftarrow$" and just write A.

Syntax

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 `:-` $B_1,\ \ldots, B_m,$ `not` $C_1,\ \ldots,$ `not` C_n `.`

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

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

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

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

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

<u>Ex.</u>: Let $P = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a, \text{not}\, b.\}$

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

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

Semantics: Interpretation

Definition: interpretation, satisfaction

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

S **satisfies** $A \leftarrow B_1, \ldots, B_m, \mathrm{not}\, C_1, \ldots, \mathrm{not}\, C_n$ 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 $P = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a, \mathrm{not}\, b.\}$

$S = \{a, b, c\}$ satisfies a, but it does not satisfy $(\mathrm{not}\, b)$.
It satisfies $c \leftarrow a, b$ because it satisfies the head because $c \in S$
It satisfies $d \leftarrow a, \mathrm{not}\, b$ because it falsifies the body because $b \in S$

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 $r \in P$.

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

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

$S_1 = \{a\}$ 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.: $P = \{a. \quad c \leftarrow a, b.\}$

$S_1 = \{a\}$ 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.: $P = \{a. \quad c \leftarrow a, b.\}$

$S_1 = \{a\}$ 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 $r \in P$.

Ex.: $P = \{a. \quad c \leftarrow a, b.\}$
$S_1 = \{a\}$ 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

Theorem: unique-model property

If P is negation-free (i.e., contains no $(\mathrm{not}\, C)$), then there is exactly one stable model, which can be computed in linear time.

Semantics without Negation – Examples

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

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

Semantics without Negation – Examples

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

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

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

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

Semantics without Negation – Examples

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

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

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

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

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

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

Semantics without Negation – Examples

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

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

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

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

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

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

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

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

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

Semantics with Negation

Definition: reduct

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

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

Semantics with Negation

Definition: reduct

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

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

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

Semantics with Negation

Definition: reduct

The **reduct** P^S of P relative to S is the least set such that
if $A \leftarrow B_1, \ldots, B_m, \mathrm{not}\, C_1, \ldots, \mathrm{not}\, C_n \in P$ 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 $(\mathrm{not}\, C) \in \mathrm{Body}(r)$ for some $C \in S$: drop the rule
- else: remove all negated literals and add to P^S

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

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

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

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

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

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

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

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

<u>Ex.</u>: $P = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a, \text{not}\, b.\}$

$$
\begin{array}{lcl}
S_1 = \{a\} & \Rightarrow & P^{S_1} = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a.\} \\
S_2 = \{a, b\} & \Rightarrow & P^{S_2} = \{a. \quad c \leftarrow a, b.\} \\
S_3 = \{a, d\} & \Rightarrow & P^{S_3} = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a.\}
\end{array}
$$

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

<u>Ex.</u>: $P = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a, \text{not}\, b.\}$

$$\begin{aligned} S_1 &= \{a\} & \Rightarrow\quad P^{S_1} &= \{a. \quad c \leftarrow a, b. \quad d \leftarrow a.\} \\ S_2 &= \{a, b\} & \Rightarrow\quad P^{S_2} &= \{a. \quad c \leftarrow a, b.\} \\ S_3 &= \{a, d\} & \Rightarrow\quad P^{S_3} &= \{a. \quad c \leftarrow a, b. \quad d \leftarrow a.\} \end{aligned}$$

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

<u>Ex.</u>: $P = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a, \text{not}\, b.\}$

$$
\begin{array}{llll}
S_1 = \{a\} & \Rightarrow & P^{S_1} = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a.\} & ✗ \\
S_2 = \{a, b\} & \Rightarrow & P^{S_2} = \{a. \quad c \leftarrow a, b.\} & \\
S_3 = \{a, d\} & \Rightarrow & P^{S_3} = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a.\} &
\end{array}
$$

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 $A \leftarrow B_1, \ldots, B_m, \mathrm{not}\, C_1, \ldots, \mathrm{not}\, C_n \in P$ 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 $(\mathrm{not}\, C) \in \mathrm{Body}(r)$ for some $C \in S$: drop the rule
- else: remove all negated literals and add to P^S

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

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

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

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

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

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

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

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

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.: $P = \{a \leftarrow \mathrm{not}\, b. \quad b \leftarrow \mathrm{not}\, a.\}$

Semantics with Negation – Examples

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

$S_1 = \{\} \qquad \Rightarrow \quad P^{S_1} =$

Semantics with Negation – Examples

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

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

Semantics with Negation – Examples

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

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

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

Semantics with Negation – Examples

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

$$S_1 = \{\} \quad \Rightarrow \quad P^{S_1} = \{a \leftarrow \cancel{\text{not}\, b}. \quad b \leftarrow \cancel{\text{not}\, a.}\}$$
$$S_2 = \{a\} \quad \Rightarrow \quad P^{S_2} = \{a \leftarrow \cancel{\text{not}\, b}. \quad \cancel{b \leftarrow \text{not}\, a.}\}$$

Semantics with Negation – Examples

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

$$
\begin{aligned}
S_1 &= \{\} &&\Rightarrow \quad P^{S_1} = \{a \leftarrow \cancel{\text{not}\, b}. \quad b \leftarrow \cancel{\text{not}\, a.}\} \\
S_2 &= \{a\} &&\Rightarrow \quad P^{S_2} = \{a \leftarrow \cancel{\text{not}\, b}. \quad \cancel{b \leftarrow \text{not}\, a.}\} \\
S_3 &= \{b\} &&\Rightarrow \quad P^{S_3} =
\end{aligned}
$$

Semantics with Negation – Examples

<u>Ex.</u>: $P = \{a \leftarrow \mathrm{not}\, b. \quad b \leftarrow \mathrm{not}\, a.\}$

$$
\begin{array}{lcl}
S_1 = \{\} & \Rightarrow & P^{S_1} = \{a \leftarrow \cancel{\mathrm{not}\, b}. \quad b \leftarrow \cancel{\mathrm{not}\, a.}\} \\
S_2 = \{a\} & \Rightarrow & P^{S_2} = \{a \leftarrow \cancel{\mathrm{not}\, b}. \quad \cancel{b \leftarrow \mathrm{not}\, a.}\} \\
S_3 = \{b\} & \Rightarrow & P^{S_3} = \{\cancel{a \leftarrow \mathrm{not}\, b.} \quad b \leftarrow \cancel{\mathrm{not}\, a}.\}
\end{array}
$$

Semantics with Negation – Examples

<u>Ex.</u>: $P = \{a \leftarrow \text{not}\, b. \quad b \leftarrow \text{not}\, a.\}$

$$
\begin{array}{llll}
S_1 = \{\} & \Rightarrow & P^{S_1} = \{a \leftarrow \cancel{\text{not}\, b}. & b \leftarrow \cancel{\text{not}\, a.}\} \\
S_2 = \{a\} & \Rightarrow & P^{S_2} = \{a \leftarrow \cancel{\text{not}\, b}. & \cancel{b \leftarrow \text{not}\, a.}\} \\
S_3 = \{b\} & \Rightarrow & P^{S_3} = \{\cancel{a \leftarrow \text{not}\, b.} & b \leftarrow \cancel{\text{not}\, a}.\} \\
S_4 = \{a, b\} & \Rightarrow & P^{S_4} =
\end{array}
$$

Semantics with Negation – Examples

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

$$
\begin{array}{lcl}
S_1 = \{\} & \Rightarrow & P^{S_1} = \{a \leftarrow \cancel{\mathrm{not}\, b.} \quad b \leftarrow \cancel{\mathrm{not}\, a.}\} \\
S_2 = \{a\} & \Rightarrow & P^{S_2} = \{a \leftarrow \cancel{\mathrm{not}\, b.} \quad \cancel{b \leftarrow \mathrm{not}\, a.}\} \\
S_3 = \{b\} & \Rightarrow & P^{S_3} = \{\cancel{a \leftarrow \mathrm{not}\, b.} \quad b \leftarrow \cancel{\mathrm{not}\, a}.\} \\
S_4 = \{a, b\} & \Rightarrow & P^{S_4} = \{\cancel{a \leftarrow \mathrm{not}\, b.} \quad \cancel{b \leftarrow \mathrm{not}\, a.}\}
\end{array}
$$

Semantics with Negation – Examples

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

$$
\begin{array}{lcl}
S_1 = \{\} & \Rightarrow & P^{S_1} = \{a. \quad b\} \\
S_2 = \{a\} & \Rightarrow & P^{S_2} = \{a.\} \\
S_3 = \{b\} & \Rightarrow & P^{S_3} = \{b.\} \\
S_4 = \{a, b\} & \Rightarrow & P^{S_4} = \{\}
\end{array}
$$

Semantics with Negation – Examples

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

$$
\begin{array}{lcl}
S_1 = \{\} & \Rightarrow & P^{S_1} = \{a. \quad b\} \qquad ✗ \\
S_2 = \{a\} & \Rightarrow & P^{S_2} = \{a.\} \\
S_3 = \{b\} & \Rightarrow & P^{S_3} = \{b.\} \\
S_4 = \{a, b\} & \Rightarrow & P^{S_4} = \{\}
\end{array}
$$

Semantics with Negation – Examples

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

$$
\begin{array}{llll}
S_1 = \{\} & \Rightarrow & P^{S_1} = \{a. \quad b\} & ✗ \\
S_2 = \{a\} & \Rightarrow & P^{S_2} = \{a.\} & ✓ \\
S_3 = \{b\} & \Rightarrow & P^{S_3} = \{b.\} & \\
S_4 = \{a, b\} & \Rightarrow & P^{S_4} = \{\} &
\end{array}
$$

Semantics with Negation – Examples

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

$$
\begin{array}{llll}
S_1 = \{\} & \Rightarrow & P^{S_1} = \{a. \quad b\} & ✗ \\
S_2 = \{a\} & \Rightarrow & P^{S_2} = \{a.\} & ✓ \\
S_3 = \{b\} & \Rightarrow & P^{S_3} = \{b.\} & ✓ \\
S_4 = \{a, b\} & \Rightarrow & P^{S_4} = \{\} &
\end{array}
$$

Semantics with Negation – Examples

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

$$
\begin{array}{llll}
S_1 = \{\} & \Rightarrow & P^{S_1} = \{a. \quad b\} & ✗ \\
S_2 = \{a\} & \Rightarrow & P^{S_2} = \{a.\} & ✓ \\
S_3 = \{b\} & \Rightarrow & P^{S_3} = \{b.\} & ✓ \\
S_4 = \{a, b\} & \Rightarrow & P^{S_4} = \{\} & ✗
\end{array}
$$

Two stable models!

Semantics with Negation – Examples

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

$$\begin{array}{llll} S_1 = \{\} & \Rightarrow & P^{S_1} = \{a. \quad b\} & ✗ \\ S_2 = \{a\} & \Rightarrow & P^{S_2} = \{a.\} & ✓ \\ S_3 = \{b\} & \Rightarrow & P^{S_3} = \{b.\} & ✓ \\ S_4 = \{a, b\} & \Rightarrow & P^{S_4} = \{\} & ✗ \end{array}$$

Two stable models!

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

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

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

Semantics with Negation – Examples

<u>Ex.</u>: $P = \{a \leftarrow \mathrm{not}\, b. \quad b \leftarrow \mathrm{not}\, a.\}$

$$
\begin{array}{llll}
S_1 = \{\} & \Rightarrow & P^{S_1} = \{a. \quad b\} & ✗ \\
S_2 = \{a\} & \Rightarrow & P^{S_2} = \{a.\} & ✓ \\
S_3 = \{b\} & \Rightarrow & P^{S_3} = \{b.\} & ✓ \\
S_4 = \{a, b\} & \Rightarrow & P^{S_4} = \{\} & ✗
\end{array}
$$

Two stable models!

<u>Ex.</u>: $P = \{a \leftarrow \mathrm{not}\, a.\}$

$$S_1 = \{\} \quad \Rightarrow \quad P^{S_1} = \{a \leftarrow \cancel{\mathrm{not}\, a}.\}$$

Semantics with Negation – Examples

<u>Ex.</u>: $P = \{a \leftarrow \text{not}\, b. \quad b \leftarrow \text{not}\, a.\}$

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

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

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

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

Two stable models!

<u>Ex.</u>: $P = \{a \leftarrow \text{not}\, a.\}$

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

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

Semantics with Negation – Examples

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

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

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

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

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

Two stable models!

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

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

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

Semantics with Negation – Examples

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

$$
\begin{array}{llll}
S_1 = \{\} & \Rightarrow & P^{S_1} = \{a. \quad b\} & ✗ \\
S_2 = \{a\} & \Rightarrow & P^{S_2} = \{a.\} & ✓ \\
S_3 = \{b\} & \Rightarrow & P^{S_3} = \{b.\} & ✓ \\
S_4 = \{a, b\} & \Rightarrow & P^{S_4} = \{\} & ✗
\end{array}
$$

Two stable models!

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

$$
\begin{array}{lll}
S_1 = \{\} & \Rightarrow & P^{S_1} = \{a.\} \\
S_2 = \{a\} & \Rightarrow & P^{S_2} = \{\}
\end{array}
$$

Semantics with Negation – Examples

<u>Ex.</u>: $P = \{a \leftarrow \text{not}\, b. \quad b \leftarrow \text{not}\, a.\}$

$$
\begin{array}{lcll}
S_1 = \{\} & \Rightarrow & P^{S_1} = \{a. \quad b\} & ✗ \\
S_2 = \{a\} & \Rightarrow & P^{S_2} = \{a.\} & ✓ \\
S_3 = \{b\} & \Rightarrow & P^{S_3} = \{b.\} & ✓ \\
S_4 = \{a, b\} & \Rightarrow & P^{S_4} = \{\} & ✗
\end{array}
$$

Two stable models!

<u>Ex.</u>: $P = \{a \leftarrow \text{not}\, a.\}$

$$
\begin{array}{lcll}
S_1 = \{\} & \Rightarrow & P^{S_1} = \{a.\} & ✗ \\
S_2 = \{a\} & \Rightarrow & P^{S_2} = \{\} &
\end{array}
$$

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

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

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

No stable model!

Semantics: Overview

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

Definition: stable model

If P contains no $(\text{not}\, C)$:
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 $(\text{not}\, C)$:
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.: $P = \{a \leftarrow a. \quad b \leftarrow \mathrm{not}\, a.\}$

S $\qquad$ P^S $\qquad$ Stable model?

Ex.: $P = \{a \leftarrow \mathrm{not}\, b. \quad b \leftarrow \mathrm{not}\, c.\}$

S $\qquad$ P^S $\qquad$ Stable model?

Example on paper