Answer Set Programming[1]

COMP441

[1] Slides designed by Christoph Schwering

$\forall x\,(\mathrm{Car}(x) \rightarrow \neg\mathrm{Entry}(x))$

$\forall x\,(\mathrm{Car}(x) \rightarrow \neg\mathrm{Entry}(x))$
$\forall x\,(\mathrm{Car}(x) \wedge \mathrm{Auth}(x) \rightarrow \mathrm{Entry}(x))$

$$\left.\begin{array}{l}\forall x\,(\mathrm{Car}(x) \to \neg\mathrm{Entry}(x)) \\ \forall x\,(\mathrm{Car}(x) \wedge \mathrm{Auth}(x) \to \mathrm{Entry}(x))\end{array}\right\} \models \mathrm{Car}(C) \wedge \mathrm{Auth}(C) \to \neg\mathrm{Entry}(C)$$

ASP at a Glance

- ASP $=$ Answer Set Programming
  - ASP $\neq$ Microsoft's Active Server Pages

- ASP belongs to logic programming
  - Like Prolog: Head $\quad$ Body or $\mathit{Head}\ :\!-\ \mathit{Body}\,.$

- 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 $\mathrm{NP}^{\mathrm{NP}}$

Motivation for ASP and this Lecture

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

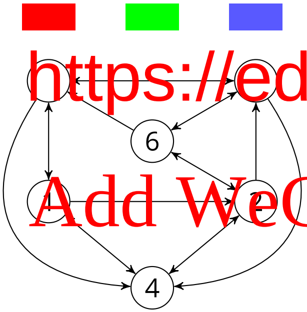
-
  - Small, simple core language
  - Great expressivity by reduction to co

- Knowing the theory is essential

Example: Graph Colouring

Definition: graph colouring problem

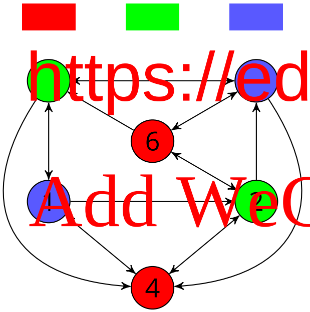
Input: graph with vertices V and edges $E \subseteq V \times V$, set of colors C.
Is there a mapping $m : V \to C$ with $m(x) \neq m(y)$ for all $(x, y) \in E$?



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 $(x, y) \in E$?



Example: Graph Colouring

Definition: graph colouring problem

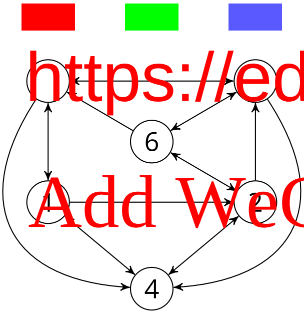
Input: graph with vertices V and edges $E \subseteq V \times V$, set of colors C.
Is there a mapping $m : V \to C$ with $m(x) \neq m(y)$ for all $(x, y) \in E$?

- Graph Coulouring is NP-complete
-
  - Mapping (neighbouring countries t
  - Compilers (register allocation)
  - Scheduling (e.g., conflicting jobs)
  - 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 $m : V \to C$ with $m(x) \neq m(y)$ for all $(x, y) \in E$?

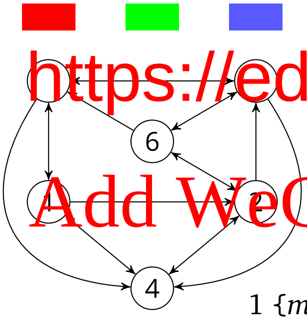


$e($
$e($
$e($
$e($

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 $(x, y) \in E$?



$e($
$e($
$e($
$e($


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

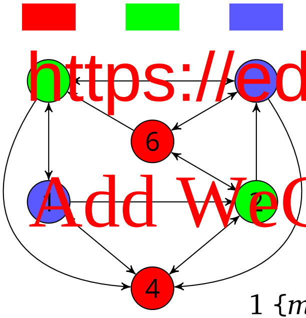
guess mapping m
verify $m(X) \neq m(Y)$

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 $(x, y) \in E$?



$e($
$e($
$e($
$e($

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

:- $e(X,Y)$, $m(X,C)$, $m(Y,C)$.   verify $m(X) \neq m(Y)$

Applications of ASP

- Automated product configuration
- Linux package manager

- Several implementations are availabl
- For this lecture: **Clingo** `www.pota`

- **Semantics of ASP programs**

- 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:

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:

Prolog vs ASP

Consider the following logic program:

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

- Prolog proves by SLD resolution:

  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:

  Algorithm defines what Prolog does
- What is the *semantics* of this logic pro

Prolog vs ASP

Consider the following logic program:

- $a.$ $\qquad$ a
  $c \leftarrow a, b.$ $\qquad$ $a \wedge b \rightarrow c$
  $d \leftarrow a, \text{not } b.$ $\qquad$ $a \wedge \neg b \rightarrow d$

- Prolog proves by SLD resolution:

  Algorithm defines what Prolog does

- What is the *semantics* of this logic pro

  - Models: $M_1 =$

a	b	c	d
1	0	0	1

1	1	1	0

$\ldots$

Prolog vs ASP

Consider the following logic program:

- $a.$ $\qquad$ a

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

  $d \leftarrow a, \text{not } b.$ $\qquad$ $a \wedge \neg b \rightarrow d$

- Prolog proves by SLD resolution:

  Algorithm defines what Prolog does
- What is the *semantics* of this logic pro

  - Models: $M_1 =$

a	b	c	d
1	0	0	1

1	1	1	0

$\cdots$

  - M_1 corresponds to Prolog, what is special about M_1?

Prolog vs ASP

Consider the following logic program:

- $a.$ $\quad\quad\quad\quad\quad\quad\quad\quad$ a
  $c \leftarrow a, b.$ $\quad\quad\quad\quad\quad$ $a \wedge b \rightarrow c$
  $d \leftarrow a, \text{not } b.$ $\quad\quad\quad$ $a \wedge \neg b \rightarrow d$

- Prolog proves by SLD resolution:

  Algorithm defines what Prolog does

- What is the *semantics* of this logic pro

  - Models: $M_1 =$

a	b	c	d
1	0	0	1

1	1	1	0

$\cdots$

  - 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
-

Intuition

The motivating guidelines behind stable model semantics are:

- A stable model satisfies all the rules of a logic program
-

Next: formalisation of this intuition

For now: only ground programs, i.e., no variab

Syntax

Definition: normal logic program (NLP)

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

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

whe

Whe

Syntax

Definition: normal logic program (NLP)

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

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

whe

Whe

For such a rule r, we define:

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

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 Eng

- dy
- S falsifies body iff S falsifies some

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 Eng

- dy
- S falsifies body iff S falsifies some

Ex.: Let $P = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a, \mathrm{no}$

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 Eng

- dy
- S falsifies body iff S falsifies some

Ex.: Let $P = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a, \text{no}$

$S = \{a, b, c\}$ 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 Eng

- dy
- S falsifies body iff S falsifies some

Ex.: Let $P = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a, \text{no}$

$S = \{a, b, c\}$ 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** $A \leftarrow B_1, \ldots, B_m, \text{not}\, C_1, \ldots, \text{not}\, C_n$ iff

A

In Eng

- dy
- S falsifies body iff S falsifies some

<u>Ex.</u>: Let $P = \{a. \quad c \leftarrow a, b. \quad d \leftarrow a, \text{no}$

$S = \{a, b, c\}$ 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 **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.:

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.:
$S_1 =$

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.:

$S_1 =$

$S_2 = \{a, b\}$ 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.:
$S_1 =$

$S_2 = \{a, b\}$ is not a stable model of P
$S_3 = \{a, b, c\}$ is not a stable model of

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.:

$S_1 =$

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

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

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.

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 $S^{i+1} = S^i$

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 $S^{i+1} = S^i$

<u>Ex.</u>:

$S^0 =$

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.:
$S^0 =$

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.:

$S^0 =$

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

$S^0 = \{\}$ 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 Eng

-
-

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 Eng

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

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

In Eng

-
-

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

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

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 Eng

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

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

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

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 Eng

-
-

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

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

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

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

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 Eng

-
-

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

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

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

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

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 Eng

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

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

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

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

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 Eng

-
-

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

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

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

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

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 Eng

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

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

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

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

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 Eng

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

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

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

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

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

Semantics with Negation – Examples

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

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

Semantics with Negation – Examples

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

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

Semantics with Negation – Examples

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

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

$S_2 = \ a \qquad\qquad P^{S_2} =$

Semantics with Negation – Examples

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

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

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

Semantics with Negation – Examples

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

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

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

$S_3 =$

Semantics with Negation – Examples

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

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

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

$S_3 =$

Semantics with Negation – Examples

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

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

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

$S_3 =$

$S_4 =$

Semantics with Negation – Examples

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

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

$S_2 = \; a \qquad P^{S_2} = \; a \quad \cancel{\text{not}\, b}. \quad \cancel{b \quad \text{not}\, a}.$

$S_3 =$

$S_4 =$

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

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

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

$S_3 =$

$S_4 =$

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

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

$S_2 = \ a \qquad\qquad P^{S_2} = \ a.$

$S_3 =$

$S_4 =$

Semantics with Negation – Examples

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

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

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

$S_3 =$

$S_4 =$

Semantics with Negation – Examples

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

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

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

$S_3 =$ ✓

$S_4 =$

Semantics with Negation – Examples

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

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

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

$S_3 =$ ✓

$S_4 =$ ✗

Two s

Semantics with Negation – Examples

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

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

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

$S_3 =$ ✓

$S_4 =$ ✗

Two s

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

Semantics with Negation – Examples

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

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

$S_2 = \ a \qquad P^{S_2} = \ a.$ ✓

$S_3 =$ ✓

$S_4 =$ ✗

Two s

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

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

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

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

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

$S_3 =$ ✓

$S_4 =$ ✗

Two s

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

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

$S_2 = \ a \qquad P^{S_2} = \ a.$ ✓

$S_3 =$ ✓

$S_4 =$ ✗

Two s

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

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

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

Semantics with Negation – Examples

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

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

$S_2 = \ a \qquad P^{S_2} = \ a.$ ✓

$S_3 =$ ✓

$S_4 =$ ✗

Two s

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

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

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

Semantics with Negation – Examples

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

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

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

$S_3 =$ ✓

$S_4 =$ ✗

Two s

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

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

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

$S_3 =$ ✓

$S_4 =$ ✗

Two s

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

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

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

$S_3 =$ ✓

$S_4 =$ ✗

Two s

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

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

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

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

Defi

If P c

S is a

S is a minimal set (w.r.t. $\subseteq$) that satis

If P contains (not C):

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

Theorem: necessary satisfaction condition

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

Semantics – Examples

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

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

Ex.:

S $\qquad$ P^S $\qquad$ el?

Example on paper

Overview of the Lecture

- Semantics of ASP programs
-
-
- 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 / S or some C_j S.

P^S c $_n \notin S$.

Integrity Constraints

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 S.

P^S c $_n \notin S$.

The

Let P' be like P except that every integrity c

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

is replaced with

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

for some new atom $dummy$.

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.

The

A choi $2k + 1$

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.

The

A choi $2k+1$

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: $min\ \{A_1,\ldots,A_k\}\ max$
  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

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

Negation in the Rule Head

Definition: rules with negated head

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

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

The

Let P

$$1 \qquad m \qquad 1 \qquad n$$

is replaced with

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

and

$$dummy \leftarrow \mathrm{not}\,A$$

for some new atom $dummy$.
Then P and P' have the same stable models (modulo dummy propositions).

Complexity

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)

The

Is S a stable model of P? – **Linear time**

Does P have a stable model? – **NP-co**

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

- Semantics of ASP programs

- ASP as modelling language

Programs with Variables

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

Programs with Variables

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

P

Programs with Variables

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

- n P

- ASP **grounds** variables with Herbra
  - Unlike Prolog: instantiation instead
  - Caution: the ground program may gr
  - Caution: function symbols make grounding Turing-complete
  - If P is finite and mentions only constants, grounding is finite

Programs with Variables

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

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

- Semantics of ASP programs

- **ASP as modelling language**

ASP Modelling

```
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).

1 {m
:- v(X).
, m(Y,C).
```

Typical ASP structure:

- Problem **instance**: a set of facts
- Problem **class**: a set of rules

Ideal modeling is **uniform**: problem clas
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.
-

Example: Non-monotonic Reasoning

Tweety the penguin:

- (Normal) Birds fly.
- Penguins are abnormal.
- Tweety is a bird. So Tweety flies.
-

$U =$

$P = \{ \ (\ ) \leftarrow \ \ (\ ), \qquad (\ ). \quad (\ ) \leftarrow \ \ (\ ). \quad (\ ).$

Example: Non-monotonic Reasoning

Tweety the penguin:

- (Normal) Birds fly.
- Penguins are abnormal.
- Tweety is a bird. So Tweety flies.
-

$U =$

$P = \{\ (\ ) \leftarrow\ (\ ),\quad (\ ).\quad (\ ) \leftarrow\ (\ ).\quad (\ ).$

$S_1 = \{b(t), f(t)\} \Rightarrow P^{S_1} = \{f(t) \leftarrow \quad . \ b(t).\}$ ✓

$S_2 = \{a(t), b(t), p(t)\} \Rightarrow P^{S_2} = \{f(t) \leftarrow \quad . \ \underline{b(t)}.\}$ ✗

Tweety flies!

Example: Non-monotonic Reasoning

Tweety the penguin:

- (Normal) Birds fly.
- Penguins are abnormal.
- Tweety is a bird. ~~So Tweety flies.~~
-

$U =$

$P = \{\ (\ ) \leftarrow \ (\ ), \quad (\ ). \quad (\ ) \leftarrow \ (\ ). \quad (\ ).$

$S_1 = \{b(t), f(t)\} \Rightarrow P^{S_1} = \{f(t) \leftarrow \quad . \ b(t).\}$ ✓

$S_2 = \{a(t), b(t), p(t)\} \Rightarrow P^{S_2} = \{f(t) \leftarrow \quad . \ \underline{b(t)}.\}$ ✗

Tweety flies!

$S_1 = \{b(t), f(t)\} \Rightarrow (P \cup \{p(t).\})^{S_1} = P_2^{S_1} \cup \{p(t).\}$ ✗

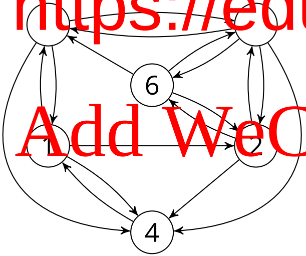
$S_2 = \{a(t), b(t), p(t)\} \Rightarrow (P \cup \{p(t).\})^{S_2} = P_2^{S_1} \cup \{p(t).\}$ ✓

Tweety doesn't fly.

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?



$r(Y)$

$\leftarrow 2$

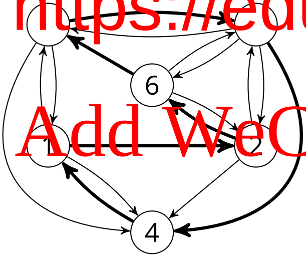
$\leftarrow 2$

$\leftarrow$ no ( ), ( ).

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?



$r(Y)$

$\leftarrow 2$

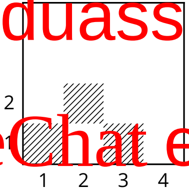
$\leftarrow 2$

$\leftarrow$ no $(\ \ ),\ (\ \ ).$

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.

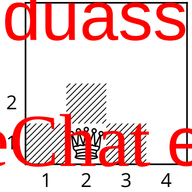


Program on paper

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