Answer Set Programming

(4) ASP as modelling language

Abdallah Saffidine

COMP4418

Overview of the Lecture

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

ASP Modelling

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

Typical ASP structure:

- Problem **instance**: a set of facts
- Problem **class**: a set of rules
  - Generator rules: often choice rules $1\ \{m(X,C) : c(C)\}\ 1 \text{ :- } v(X).$
  - Test rules: often integrity constraints $\text{:- } e(X,Y), m(X,C), m(Y,C).$

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.

$$U = \{f(X) \leftarrow b(X), \mathrm{not}\, a(X). \quad a(X) \leftarrow p(X). \quad b(t).\}$$
$$P = \{f(t) \leftarrow b(t), \mathrm{not}\, a(t). \quad a(t) \leftarrow p(t). \quad b(t).\}$$

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.

$$U = \{f(X) \leftarrow b(X), \mathrm{not}\, a(X). \quad a(X) \leftarrow p(X). \quad b(t).\}$$
$$P = \{f(t) \leftarrow b(t), \mathrm{not}\, a(t). \quad a(t) \leftarrow p(t). \quad b(t).\}$$

$$S_1 = \{b(t), f(t)\} \quad \Rightarrow \quad P^{S_1} = \{f(t) \leftarrow b(t)\cancel{, \mathrm{not}\, a(t)}. \quad a(t) \leftarrow p(t). \quad b(t).\} \checkmark$$
$$S_2 = \{a(t), b(t), p(t)\} \quad \Rightarrow \quad P^{S_2} = \{\cancel{f(t) \leftarrow b(t), \mathrm{not}\, a(t).} \quad a(t) \leftarrow p(t). \quad b(t).\} ✗$$

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.

$$U = \{f(X) \leftarrow b(X), \mathrm{not}\, a(X). \quad a(X) \leftarrow p(X). \quad b(t).\}$$
$$P = \{f(t) \leftarrow b(t), \mathrm{not}\, a(t). \quad a(t) \leftarrow p(t). \quad b(t).\}$$

$$S_1 = \{b(t), f(t)\} \quad \Rightarrow \quad P^{S_1} = \{f(t) \leftarrow b(t)\cancel{, \mathrm{not}\, a(t)}. \quad a(t) \leftarrow p(t). \quad b(t).\} \checkmark$$
$$S_2 = \{a(t), b(t), p(t)\} \quad \Rightarrow \quad P^{S_2} = \{\cancel{f(t) \leftarrow b(t), \mathrm{not}\, a(t).} \quad a(t) \leftarrow p(t). \quad b(t).\} \text{✗}$$

Tweety flies!

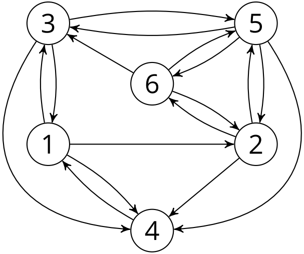
$$S_1 = \{b(t), f(t)\} \quad \Rightarrow \quad (P \cup \{p(t).\})^{S_1} = P_2^{S_1} \cup \{p(t).\} \quad \text{✗}$$
$$S_2 = \{a(t), b(t), p(t)\} \quad \Rightarrow \quad (P \cup \{p(t).\})^{S_2} = P_2^{S_1} \cup \{p(t).\} \quad \checkmark$$

Tweety doesn't fly.

Example: Hamilton Cycle

Definition: Hamilton cycle problem

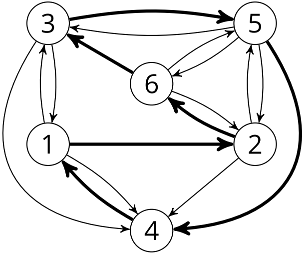
Input: graph with vertex set V and edges $E \subseteq V \times V$.
Is there a cycle that visits every vertex exactly once?



Example: Hamilton Cycle

Definition: Hamilton cycle problem

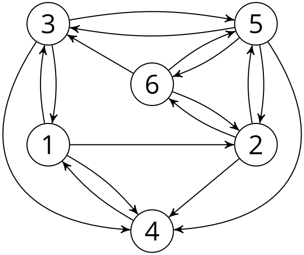
Input: graph with vertex set V and edges $E \subseteq V \times V$.
Is there a cycle that visits every vertex exactly once?



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?

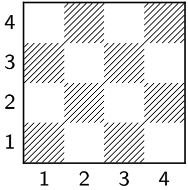


$$\{p(X,Y)\} \leftarrow e(X,Y).$$
$$r(X) \leftarrow p(1,X).$$
$$r(Y) \leftarrow r(X), p(X,Y).$$
$$\leftarrow 2\ \{p(X,Y)\}\ , v(X).$$
$$\leftarrow 2\ \{p(X,Y)\}\ , v(Y).$$
$$\leftarrow \text{not}\, r(X), v(X).$$

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.

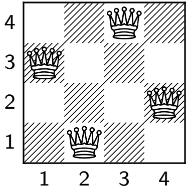


Program on paper

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