Artificial Intelligence II

2013/2014 - Prof: Daniele Nardi, Joachim Hertzberg

Exercitation 2 - Roberto Capobianco

Non-Monotonic Reasoning

Closed World Assumption (Recap)

- If something is not in the KB, it can reasonably be assumed to be false.
- Negative information is added to the KB in order to create KB^+ (closure of the KB).
- Queries are directly answered on KB^+.
- CWA produces a complete KB (i.e., for all *a* KB^+ entails either *a* or ¬*a*).

CWA Example (1)

- KB = {distracted(fabio), attends(fabio), attends(antonio), $\forall x \neg distracted(x) \wedge attends(x) \rightarrow learns(x)$}

- Does CWA entail learns(antonio)?

CWA Example (1-sol)

- KB closure:

$$\mathrm{KB}^{+} = \mathrm{KB} \cup \{\neg \mathrm{distracted}(\mathrm{antonio})\}$$

- $\mathrm{KB}^{+} \models \mathrm{learns}(\mathrm{antonio})$

- Domain closure:

$$\mathrm{KB}^{\diamond} = \mathrm{KB}^{+} \cup \{\forall x\, [x = antonio \vee x = fabio]\}$$

- $\mathrm{KB}^{\diamond} \models \forall x\, \mathrm{attends}(x)$

CWA Inconsistency

$$KB = \{attends(fabio) \vee attends(antonio)\}$$

- KB does not contain attends(fabio), therefore KB^+ contains $\neg attends(fabio)$
- KB does not contain attends(antonio), therefore KB^+ contains $\neg attends(antonio)$

$$KB^+ = KB \cup \{\neg attends(fabio), \neg attends(antonio)\}$$

KB^+ is inconsistent!

Generalized CWA (Recap)

- ¬L is added to KB^* (closure of the KB) if:
  - L is a positive literal;
  - There is no positive clause C such that KB ⊨ L ∨ C and KB does not entail C;

Generalized CWA Example

$$KB = \{attends(fabio) \vee attends(antonio)\}$$

- Generalized CWA closure:

$$KB^* = \{attends(fabio) \vee attends(antonio)\}$$

Negation in Logic Programming

- Negation as failure can be formalized through CWA.

- **Problem:** the behavior of SLDNF resolution used by Prolog, in the presence of negation in the bodies of rules, does not fully match the truth tables of classical propositional logic.

Negation and Truth Assignments (1)

```
p.
r :- p, q.
s :- p, not q.
```

Prolog: `p` is True, `q` is False, `r` is False, `s` is True.

The rules of the given program can be viewed as propositional formulas. The previous truth assignment is a model of the program, since each rules gets the value True. The following one is also a model:

`p` is True, `q` is True, `r` is True, `s` is False.

Negation and Truth Assignments (2)

`p` is True, `q` is False, `r` is False, `s` is True.

The above assignment is special in the sense that:

- It is a model;
- It correctly represents the behavior of SLDNF resolution.

Indeed, it is a stable model.

Stable Models (Recap)

- Given a ground program denoted as ground(π), choose a model M;
- Compute the reduct of ground(π), by removing all the clauses that contain the negated of a literal in M and all the negative literals that do not appear in M;
- Verify if the minimal Herbrand model of the reduct of ground(π) equals M;

Stable Models Example (1)

- Consider the program π:

$$p(1, 2)$$
$$q(x) \leftarrow p(x, y), \neg q(y)$$

- We replace the second rule with its ground instances:

$$q(1) \leftarrow p(1, 1), \neg\, q(1)$$
$$q(1) \leftarrow p(1, 2), \neg\, q(2)$$
$$q(2) \leftarrow p(2, 1), \neg\, q(1)$$
$$q(2) \leftarrow p(2, 2), \neg\, q(2)$$

Stable Models Example (2)

- Let $M = \{q(2)\}$. Then the reduct π_M is

$$p(1, 2)$$
$$q(1) \leftarrow p(1, 1)$$
$$q(2) \leftarrow p(2, 1)$$

- The minimal Herbrand model is $\{p(1, 2)\}$

M is not stable! (M is not a model of π)

Stable Models Example (3)

- Let's try again! $M' = \{p(1, 2), q(1)\}$ (M' is a model of π)
- $\pi_{M'}$ is:

$$p(1, 2)$$
$$q(1) \leftarrow p(1, 2)$$
$$q(2) \leftarrow p(2, 2)$$

- The minimal Herbrand model of this program is $\{p(1, 2), q(1)\}$

Hence, M' is stable!

Stable Models Exercise (1)

$$\pi = \{\text{Employee}(d) \wedge \neg\text{Sick}(d) \rightarrow \text{Day}(f_g(d)),$$
$$\text{Employee}(d) \wedge \neg\text{Sick}(d) \rightarrow \text{WorksOn}(d, f_g(d)),$$
$$\text{Employee}(\text{Bianchi}), \text{Employee}(\text{Rossi}),$$
$$\text{Sick}(\text{Bianchi}), \neg\text{Sick}(\text{Rossi})\}$$

Are there stable models?

Stable Models Exercise (2)

$$\text{ground}(\pi) =$$

$$\{\text{Employee(Rossi)} \wedge \neg\text{Sick(Rossi)} \rightarrow \text{Day}(f_g(\text{Rossi})),$$

$$\text{Employee(Rossi)} \wedge \neg\text{Sick(Rossi)} \rightarrow \text{WorksOn(Rossi, } f_g(\text{Rossi})),$$

$$\text{Employee(Bianchi)} \wedge \neg\text{Sick(Bianchi)} \rightarrow \text{Day}(f_g(\text{Bianchi})),$$

$$\text{Employee(Bianchi)} \wedge \neg\text{Sick(Bianchi)} \rightarrow \text{WorksOn(Bianchi, } f_g(\text{Bianchi})),$$

$$\text{Employee(Bianchi), Employee(Rossi), Sick(Bianchi), } \neg\text{Sick(Rossi)}\}$$

Stable Models Exercise (3)

- ground(π) admits as a model:

$$M = \{\text{Employee(Bianchi)}, \text{Employee(Rossi)}, \text{Sick(Bianchi)}, \text{WorksOn(Rossi, } f_g\text{(Rossi))}, \text{Day(}f_g\text{(Rossi))}\}$$

- π_M is the reduct of ground(π)

$$\pi_M = \{\text{Employee(Rossi)} \rightarrow \text{Day(}f_g\text{(Rossi))}, \text{Employee(Rossi)} \rightarrow \text{WorksOn(Rossi, } f_g\text{(Rossi))}, \text{Employee(Bianchi)}, \text{Employee(Rossi)}, \text{Sick(Bianchi)}\}$$

Stable Models Exercise (4)

- ground(π) admits as a model:

$$M = \{\text{Employee(Bianchi)}, \text{Employee(Rossi)}, \text{Sick(Bianchi)}, \text{WorksOn(Rossi, } f_g(\text{Rossi})), \text{Day}(f_g(\text{Rossi}))\}$$

- π_M is the reduct of ground(π)

$$\pi_M = \{\text{Employee(Rossi)} \rightarrow \text{Day}(f_g(\text{Rossi})),$$
$$\text{Employee(Rossi)} \rightarrow \text{WorksOn(Rossi, } f_g(\text{Rossi})),$$
$$\text{Employee(Bianchi)}, \text{Employee(Rossi)}, \text{Sick(Bianchi)}\}$$

M is stable!

Answer Set Programming

- In Answer Set Programming, search problems are reduced to computing stable models, and answer set solvers (i.e., programs for generating stable models) are used to perform search.
- ASP systems solve a problem in two steps.
  - A **grounder** "grounds" the problem file(s) to an intermittent format. Note: Those can be very large files.
  - A **solver** reads this grounded file and - hopefully - solves the problem by generating answers sets (solutions).
- Potassco (the Potsdam Answer Set Solving Collection) tools which includes the grounder **gringo**, the solver **clasp**

```
sudo apt-get install gringo clasp
```

ASP Example

- ASP follows the syntax of Prolog: variables begin with upper case letters and constants begin with lower case letters.

- Write program.lp

```
p.
r :- p, q.
s :- p, not q.
```

- Run `gringo program.lp | clasp`

- You will get the following result:

```
Answer: 1
p s
SATISFIABLE
```

Circumscription (1)

- CWA minimizes all predicates, Circumscription only minimizes predicates that model abnormality (e.g., Ab(x)).

$$\mathrm{KB} = \{\mathrm{Ab}(\mathrm{fabio}), \mathrm{Attends}(\mathrm{fabio}), \mathrm{Attends}(\mathrm{antonio}),$$
$$\forall x\ \neg\mathrm{Ab}(x) \wedge \mathrm{Attends}(x) \rightarrow \mathrm{Learns}(x)\}$$

- Does circumscription entail Learns(antonio)?

Circumscription (2)

- CWA minimizes all predicates, Circumscription only minimizes predicates that model abnormality (e.g., Ab(x)).

$$KB = \{Ab(fabio), Attends(fabio), Attends(antonio),$$
$$\forall x \neg Ab(x) \wedge Attends(x) \rightarrow Learns(x)\}$$

- Does circumscription entail Learns(antonio)?
- Assume P = {Ab} and take the minimal extension of Ab (i.e., {Ab(fabio)}

Yes, circumscription entails Learns(antonio)

Default Reasoning (Recap)

$<W,D>$

- W: as usual, a set of first order sentences
- D: default rules of the form

$$\alpha : \beta / \gamma$$

where:

- α is a prerequisite and β is a justification
- γ is a conclusion

- ε is an extension of a default theory $<W,D>$
- Being π a sentence

$$\pi \in \varepsilon \text{ iff } W \cup \{\gamma \mid <\alpha : \beta / \gamma> \in D, \alpha \in \varepsilon, \neg\beta \notin \varepsilon\} \models \pi$$

- Like extensions in CWA, we add default assumptions to a set of basic facts

Default Example (1)

- Consider <W,D>

$$W = \{\text{Robot(Sonny)}, \text{Robot(Polly)}, \neg\text{Worker(Polly)}\}$$

$$D = \{\text{Robot(x): Worker(x)/Worker(x)},$$
$$\text{Robot(x):}\neg\text{Payed(x)}/\neg\text{Payed(x)}$$
$$\text{Worker(x):Payed(x)/Payed(x)}\}$$

- Are Sonny and Polly payed?

Default Example (2)

- Rules 2 and 3 are in conflict, therefore there are 2 extensions

$$\varepsilon_1 = W \cup \{\text{Worker}(\text{Sonny}), \neg\text{Payed}(\text{Sonny}), \neg\text{Payed}(\text{Polly})\}$$
$$\varepsilon_2 = W \cup \{\text{Worker}(\text{Sonny}), \text{Payed}(\text{Sonny}), \neg\text{Payed}(\text{Polly})\}$$

- Skeptical conclusions Worker(Sonny) and ¬Payed(Polly)
- Additional Credolous conclusions ¬Payed(Sonny) or Payed(Sonny)

Default Exercise (1)

$$W = \{\text{Male(Neri)}, \text{Employee(Neri)}, \text{GoodImpression(Neri)}, \text{Male(Rossi)}, \text{Employee(Rossi)}, \text{GoodImpression(Rossi)}, \text{FrontOffice(Rossi)}, \forall x\ (\neg\text{FrontOffice}(x) \vee \neg\text{ITExpert}(x))\}$$

- Defaults:
  - Male employees generally do not have to wear a suit, unless they work at the front office;
  - Generally, employees that want to make a good impression wear a suit, unless they are IT experts.

Default Exercise (2)

$$D = \{Employee(x) \wedge Male(x) : \neg FrontOffice(x) / \neg Suit(x),$$
$$Employee(x) \wedge GoodImpression(x) : \neg ITExpert(x) / Suit(x)\}$$

Default Exercise (3)

$$D = \{Employee(x) \wedge Male(x) : \neg FrontOffice(x) / \neg Suit(x),$$
$$Employee(x) \wedge GoodImpression(x) : \neg ITExpert(x) / Suit(x)\}$$

$$\varepsilon 1 = W \cup \{Suit(Rossi), \neg Suit(Neri)\}$$
$$\varepsilon 2 = W \cup \{Suit(Rossi), Suit(Neri)\}$$

The order of application of default rules makes the difference

Non-Monotonic Reasoning with ASP (1)

```
fly(X) :- bird(X), not -fly(X).
bird(twittie).
-fly(X) :- penguin(X).
```

Run the program:

```
gringo program.lp | clasp
```

You will get:

```
Answer: 1
bird(twittie) fly(twittie)
SATISFIABLE
```

Non-Monotonic Reasoning with ASP (2)

- Add the line below to the end of the program.

```
penguin(twittie).
```

- Now `-fly(twittie)` is part of the answer.

Thank you!