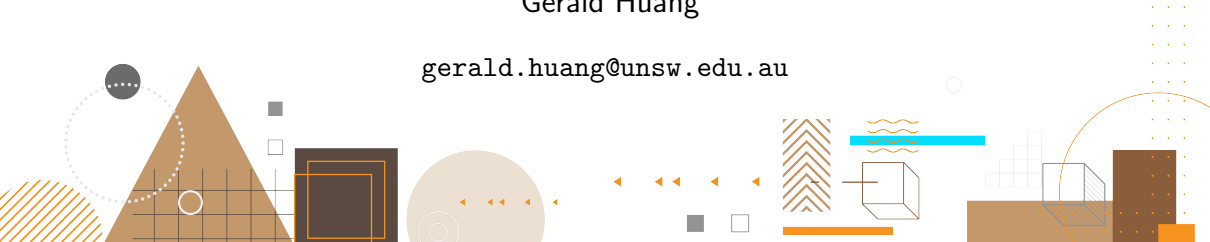
COMP4418

Knowledge Representation and Reasoning

Week 1 tutorial

Gerald Huang

gerald.huang@unsw.edu.au



Part I: Tutorial exercises

Part II: Project specification

Logic programming

A *logic program* is a program that consists of rules which define constraints on what we can compute.

- A **positive logic program** is a program that consists only of rules of the form:

$$A \leftarrow B,$$

  where B is a positive literal (or atomic proposition).

In other words, any negation is only applied to an inequality!

$$Brother(x, y) \leftarrow Parent(z, x) \wedge Parent(z, y) \wedge Male(x) \wedge (x \neq y).$$

Stable models for positive logic programs

Let S be a set of atoms. We say that S is a *stable model* of a program P if

- All of the rules of P are satisfied/consistent with the atoms in S.
- S is minimal (with respect to $\subseteq$).

Consider the program $P = \{a., c \leftarrow a, b.\}$. Then $S = \{a\}$ forms the stable model.

Do they always exist? Yes!

Exercise 1

Determine the models, and decide which models are stable.

$$P = \begin{cases} \mathit{rain} \leftarrow \\ \mathit{wet} \leftarrow \mathit{rain} \\ \mathit{wet} \leftarrow \mathit{sprinkler}. \end{cases}$$

We note that *any* model must contain *rain*. But, if $rain \in X$, then $wet \in X$. Thus the possible models are

$$X = \{rain, wet\}, \quad Y = \{rain, wet, sprinkler\}.$$

The stable model is the minimal model which is $X = \{rain, wet\}$.

Exercise 1

Determine the models, and decide which models are stable.

$$P = \begin{cases} \mathit{coffee} \leftarrow \\ \mathit{lemon} \leftarrow \mathit{tea} \\ \mathit{sugar} \leftarrow \mathit{coffee} \\ \mathit{milk} \leftarrow \mathit{coffee}, \mathit{sugar} \\ \mathit{tea} \leftarrow \mathit{lemon} \\ \mathit{tea} \leftarrow \mathit{diet} \end{cases}$$

We note that *any* model must contain *coffee*. But, if $coffee \in X$, then $sugar \in X$ which implies that $milk \in X$.

Now, if $lemon \in X$, then $tea \in X$. If $diet \in X$, then $tea \in X$ which implies that $lemon \in X$. Therefore, we have three models

$$X = \{coffee, sugar, milk\},$$
$$Y = \{coffee, sugar, milk, lemon, tea\},$$
$$Z = \{coffee, sugar, milk, lemon, tea, diet\}.$$

The stable model is $X = \{coffee, sugar, milk\}$.

Exercise 1

Determine the models, and decide which models are stable.

$$P = \begin{cases} \mathit{shirt} \leftarrow \\ \mathit{sneakers} \leftarrow \\ \mathit{pants} \leftarrow \mathit{sneakers} \\ \mathit{skirt} \leftarrow \mathit{shirt}, \mathit{sandals} \\ \mathit{sandals} \leftarrow \mathit{dress}. \end{cases}$$

We note that *any* model must contain *shirt* and *sneakers*. But this implies that $pants \in X$.

Now, if $sandals \in X$, then $skirt \in X$. If $dress \in X$, then $sandals \in X$. Therefore, the four models are

$$
\begin{aligned}
W &= \{shirt, sneakers, pants\} \\
X &= \{shirt, sneakers, pants, skirt\}, \\
Y &= \{shirt, sneakers, pants, sandals, skirt\}, \\
Z &= \{shirt, sneakers, pants, sandals, skirt, dress\}.
\end{aligned}
$$

The stable model is $W = \{shirt, sneakers, pants\}$.

Exercise 1

Determine the models, and decide which models are stable.

$$P = \begin{cases} red \leftarrow \\ meat \leftarrow cabbage \\ meat \leftarrow red \\ fish \leftarrow asparagus \\ asparagus \leftarrow fish, white \\ white \leftarrow fish. \end{cases}$$

Follow the same procedure to verify that the models are

$$
\begin{aligned}
U &= \{red, meat\},\\
V &= \{red, meat, cabbage\},\\
W &= \{red, meat, white\},\\
X &= \{red, meat, white, cabbage\},\\
Y &= \{red, meat, white, fish\},\\
Z &= \{red, meat, white, fish, asparagus\}.
\end{aligned}
$$

The stable model is $U = \{red, meat\}$.

Normal logic programs

A **normal logic program** is a program that consists of rules of the form:

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

Negations are also applied to the atomic propositions!

Question: Do stable models always exist? No!

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

Computing stable models

To compute stable models, we begin with a set of atomic propositions X; the goal here is to *partially* evaluate the rules of the program P. Then we see what the resulting simplified program's "least model" P^X looks like.

Reduct

Let X be a set of atoms. The *reduct* P^X is the set of clauses obtained from P as follows;

- delete any clause that has not A in the body where $A \in X$.
- delete every negated expression.

The result is the set of clauses that belong in the reduct P^X.

Then X is a stable model of P if and only if X is a stable model of P^X.

Exercise 2

Determine the stable models of the following normal logic programs.

$$P = \begin{cases} sprinkler \leftarrow \neg rain \\ rain \leftarrow \neg sprinkler \\ wet \leftarrow rain \\ wet \leftarrow sprinkler \end{cases}$$

Intuitively, we see that *rain* and *sprinkler* cannot be in the same model. Similarly, if $rain \in X$, then $wet \in X$; therefore, we claim that $X = \{rain, wet\}$ is a stable model. We can show this by performing the reduct P^X.

1. We delete any clause that contains $\neg A$, for some $A \in X$, in the body; this removes rule 1.
2. We delete any negated expressions.

$$P^X = \begin{cases} rain \\ wet \leftarrow rain \\ wet \leftarrow sprinkler \end{cases}$$

Stable model of P^X is $Y = \{rain, wet\}$; since $X = Y$, $X = \{rain, wet\}$ is a stable model. The other stable model is $\{sprinkler, wet\}$.

Exercise 2

Determine the stable models of the following normal logic programs.

$$P = \begin{cases} diet \leftarrow \neg sugar \\ coffee \leftarrow \neg tea \\ lemon \leftarrow tea \\ sugar \leftarrow coffee \\ milk \leftarrow coffee, sugar \\ tea \leftarrow lemon \\ tea \leftarrow diet \end{cases}$$

Intuitively, if $sugar \notin X$, then $diet \in X$ because the stable model will contain *diet* as a fact. Similarly, if $tea \notin X$, then $coffee \in X$.

Let's take $sugar \notin X$. Then $diet \in X$, which implies that $tea \in X$ and $lemon \in X$. Therefore, a stable model is $X = \{diet, tea, lemon\}$ (check this!).

Now suppose that $sugar \in X$. Then $diet \notin X$ which implies that $tea \notin X$ and $coffee \in X$. But this implies that $milk \in X$. Therefore, another stable model is $X = \{sugar, milk, coffee\}$ (check this as well!).

This exhausts all of the possibilities! Thus, our two stable models are $\{diet, tea, lemon\}, \{sugar, milk, coffee\}$.

Exercise 2

Determine the stable models of the following normal logic programs.

$$P = \begin{cases} \mathit{dress} \leftarrow \neg \mathit{shirt} \\ \mathit{shirt} \leftarrow \neg \mathit{dress} \\ \mathit{sandals} \leftarrow \neg \mathit{sneakers} \\ \mathit{sneakers} \leftarrow \neg \mathit{sandals} \\ \mathit{pants} \leftarrow \mathit{sneakers} \\ \mathit{skirt} \leftarrow \mathit{shirt}, \mathit{sandals} \\ \mathit{sandals} \leftarrow \mathit{dress} \end{cases}$$

We see that we have

$$dress \leftarrow \neg shirt, \quad sandals \leftarrow \neg sneakers.$$

Therefore, we have four possible decisions we can make (to add shirt and/or to add sneakers).

We need to make a decision for *shirt* and *sneakers*. If $shirt, sneakers \in X$, then $pants \in X$. Then $X = \{shirt, sneakers, pants\}$ forms a stable model of our program (check this!).

If $shirt \in X$ and $sneakers \notin X$, then $sandals \in X$. But this implies that $skirt \in X$. Then $X = \{shirt, sandals, skirt\}$ forms another stable model (check this!).

Now suppose that $shirt \notin X$ and $sneakers \in X$. Since $shirt \notin X$, then $dress \in X$ which implies that $sandals \in X$. But we now have $sneakers, sandals \in X$ (a contradiction!). The last stable model can be shown to be $X = \{dress, sandals\}$ (check this!).

Exercise 2

Determine the stable models of the following normal logic programs.

$$P = \begin{cases} asparagus \leftarrow \neg cabbage \\ cabbage \leftarrow \neg asparagus \\ red \leftarrow \neg white \\ meat \leftarrow cabbage \\ meat \leftarrow red \\ fish \leftarrow asparagus \\ asparagus \leftarrow fish, white \\ white \leftarrow fish \end{cases}$$

Repeat the same intuition! You have four possibilities depending on

$$asparagus \leftarrow \neg cabbage, \quad red \leftarrow \neg white.$$

Check that the stable models (using reduct!) are

$$X = \{red, meat, cabbage\}, \quad Y = \{white, fish, asparagus\}.$$

Project Specification

In this course, you will be doing a project in groups of 2-3 (or individually). Project A is to get you thinking about a project you may want to explore further in Projects B and C; therefore, it is imperative that you think long and hard about what might be a suitable project!

- The problem that you propose should be suitable to solve using KRR methods!
- Think about what constraints you want to model, what your input data is, and what your candidate solution(s) should look like.
- Typically, a suitable project would be of the *combinatorial problem* style.

What constitutes a *combinatorial problem*?

A combinatorial problem typically involves an ordering, assignment, or grouping of distinct/finite objects which satisfy certain constraints.

- Graph colouring on n vertices.
- Satisfiability of a formula.
- Allocation of classes.
- Items placed in a knapsack of finite capacity.

Part I: Overview

In this part of Project A, you want to come up with an idea of what your project would be about.

- Make sure your project overview is clear so that it's easy for the markers to understand specifically what your project is about.

Part II: Prototyping

In this part of Project A, you want to start thinking about what sort of data you need and how you'd want to *think* about modelling your problem.

- Think about how you should model your constraints.
- Think about what sort of input data you'd need.
- Think about what your solution(s) should look like!

Part III: Production and self-evaluation

In the production phase, think about who would use your project. Is it university students? Farmers? City workers?

Finally, evaluate yourself honestly about how you're tracking with in your group. Did you do enough to warrant marks?