CITS3005 Knowledge Representation Homework 1

August 25, 2023

Worth:10% of final grade
Due: 5pm, Friday August 25, 2023.
Submit: `https://secure.csse.uwa.edu.au/run/cssubmit/`

This home exercise requires you to complete three tasks demonstrating: formal reasoning in first order logic; logic programing; and and knowledge representation.

1. Consider a basic social network that consists of *users* and *posts*. The network has the following *rules*. Each user can

   (a) Each user can *follow* one or more other users.

   (b) Each user can *write* posts.

   (c) Each user can *see* posts.

   (d) Each user can *like* posts.

   (e) Every user *follows* themselves.

   (f) A user can only *like* a post if they can *see* that post.

   (g) A user can *see* a post if and only if it was *written* or *liked* by someone that they follow.

   With the solution network, the following *facts* are known:

   (a) Jack follows Jamie.

   (b) Jamie follows Jamal.

   (c) Jamal follows Jack.

   (d) Jamal posts "Hi!".

   (e) Jamie likes Jamal's post.

   Complete the following exercises:

   (a) Formalise the em rules and *facts* in first order logic.

   (b) Convert the first order logic statement into a CNF knowledge base (i.e. a conjunction of clauses).

   (c) Apply clausal resolution to show that Jack can see Jamal's post.

   3 marks

2. Consider the following statement in first order logic:

$$(\forall X \forall Y (p(X,Y) \rightarrow p(Y,X)) \wedge \exists X \forall Y p(X,Y)) \rightarrow \forall X \forall Y \exists Z (p(X,Z) \wedge p(Z,Y))$$

(a) Describe the statement in natural language and explain why it is intuitively valid.

(b) Apply the following natural deduction system to prove that it is valid:

true	$\frac{}{\top}$	false	$\frac{\alpha \wedge \neg\alpha}{\bot}$
$\neg$-intro	$\frac{\alpha \rightarrow \bot}{\neg\alpha}$	$\neg$-elim	$\frac{\neg\neg\alpha}{\alpha}$
$\wedge$-intro	$\frac{\alpha \quad \beta}{\alpha \wedge \beta}$	$\wedge$-elim	$\frac{\alpha \wedge \beta}{\alpha \quad \beta}$
$\rightarrow$ -intro	$\frac{\begin{array}{c}\alpha \\ \vdots \\ \beta\end{array}}{\alpha \rightarrow \beta}$	$\rightarrow$ -elimination	$\frac{\alpha \quad \alpha \rightarrow \beta}{\beta}$
$\exists$-intro*	$\frac{\alpha(t)}{\exists X \alpha(X)}$	$\exists$-elim†	$\frac{\beta(t) \rightarrow \alpha}{\exists X \beta(X) \rightarrow \alpha}$
$\forall$-intro†	$\frac{\alpha \rightarrow \beta(t)}{\alpha \rightarrow \forall X \beta(X)}$	$\forall$-elim*	$\frac{\forall X \alpha(X)}{\alpha(t)}$

*: here t cannot occur within the scope of a quantifier over any sub-term of t (t must be *free* in α).

†: here, α cannot contain the term t or any of its sub-terms, and t cannot be contained within an undischanged assumption.

3 marks

3. Write a logic program to propose solutions to Wordle problems (see `https://www.nytimes.com/games/wordle/index.html`).

Wordle consists of a secret five letter word that players must try to guess. Each guess gives the player feedback by highlighting: which letters are in the word, and in the right place; and which letters are in the word but in the wrong place. (We will assume that no candidate word have repeated letters).

Write a logic program to help solve Wordle problems. The logic program will consist of a set of words (which are all the possible candidates for the secret word), as well as the predicates:

- `green(W,I,C)`: which is true if word `W` contains character `C` at index `I`.
- `orange(W,I,C)`: which is true if word `W` contains character `C`, but not at index `I`.
- `grey(W,C)`: which is true if word `W` does not contain character `C`.

You may define other helper predicates if you like.

An example query is:

```
?- word(X), green(X,3,'C'),orange(X,2,'L'), grey(X,'I'), show_string(X).
BLACK
```

Note, prolog treats strings as arrays of character codes, show the `show_string` predicate uses `char_code` to write the string.

4 marks