491 Knowledge Representation

Tutorial Exercise

Logic databases (Revision)

SOLUTIONS

Question 1

- William frequents every bar that serves at least one of the beers he likes.

$$\forall b\,[\,\textit{William frequents } b \;\leftarrow\; \exists d\,(\,\textit{William likes } d \;\wedge\; b \textit{ serves } d\,)\,]$$

or equivalently

$$\forall b\,\forall d\,[\,\textit{William frequents } b \;\leftarrow\; (\,\textit{William likes } d \;\wedge\; b \textit{ serves } d\,)\,]$$

Why equivalently? Because $\forall x\,(\,p(x) \;\leftarrow\; \exists y\,q(x,y)\,)$ is logically equivalent to $\forall x\,\forall y\,(\,p(x) \;\leftarrow\; q(x,y)\,)$.

- Harry frequents any bar that does not serve a beer he does not like.

$$\forall b\,[\,\textit{Harry frequents } b \;\leftarrow\; \neg\exists d\,(\,b \textit{ serves } d \;\wedge\; \neg\textit{Harry likes } d\,)\,]$$

which is logically equivalent to

$$\forall b\,[\,\textit{Harry frequents } b \;\leftarrow\; \forall d\,(\,b \textit{ serves } d \;\rightarrow\; \textit{Harry likes } d\,)\,]$$

- Charles frequents every bar that serves all of the beers he likes.

$$\forall b\,[\,\textit{Charles frequents } b \;\leftarrow\; \forall d\,(\,\textit{Charles likes } d \;\rightarrow\; b \textit{ serves } d\,)\,]$$

which is logically equivalent to

$$\forall b\,[\,\textit{Charles frequents } b \;\leftarrow\; \neg\exists d\,(\,\textit{Charles likes } d \;\wedge\neg\; b \textit{ serves } d\,)\,]$$

- Camilla frequents every bar that Charles frequents, and also any bar that serves *Young's Special Bitter*.

$$\forall b\,(\,\textit{Camilla frequents } b \;\leftarrow\; \textit{Charles frequents } b\,) \;\wedge\; \forall b\,(\,\textit{Camilla frequents } b \;\leftarrow\; b \textit{ serves Young's}\,)$$

which is logically equivalent to

$$\forall b\,(\,\textit{Camilla frequents } b \;\leftarrow\; (\textit{Charles frequents } b \;\vee\; b \textit{ serves Young's})\,)$$

Question 2 You can declare the infix operators in Prolog like this:

```
:-  op(550, xfx, [likes, serves, frequents]).
```

Then:

- $\forall b\,\forall d\,[\,\textit{William frequents } b \;\leftarrow\; (\,\textit{William likes } d \;\wedge\; b \textit{ serves } d\,)\,]$

```
'William' frequents B :-
    'William' likes D, B serves D.
```

- $\forall b\,[\,\textit{Harry frequents } b \;\leftarrow\; \neg\exists d\,(\,b \textit{ serves } d \;\wedge\; \neg\textit{Harry likes } d\,)\,]$

```
'Harry' frequents B :-
    \+ (B serves D, \+ 'Harry' likes D).
```

But:

(1) In what sense is Prolog's negation-by-failure `\+` a correct representation of truth-functional negation $\neg$? We are coming to that.

(2) Prolog can't be used to generate bindings from negation-by-failure calls. So we need something like:

```
'Harry' frequents B :-
    B serves _,   % generates the name of a bar B
    \+ (B serves D, \+ 'Harry' likes D).
```

(3) Some Prologs (and other logic programming/deductive database implementations) don't allow nested negation-as-failure. So then we have to write something like this:

```
'Harry' frequents B :-
    B serves _,   % generates the name of a bar B
    \+ bad_for_Harry(B).
bad_for_Harry(B) :-
   B serves D, \+ 'Harry' likes D.
```

- $\forall b\,[\,\textit{Charles frequents } b \;\leftarrow\; \forall d\,(\,\textit{Charles likes } d \;\rightarrow\; b \textit{ serves } d\,)\,]$

```
'Charles' frequents B :-
    B serves _,   % generates the name of a bar B
    \+ ('Charles' likes D, \+ B serves D).
```

- $\forall b\,(\,\textit{Camilla frequents } b \;\leftarrow\; (\textit{Charles frequents } b \;\vee\; b \textit{ serves Young's})\,)$

```
'Camilla' frequents B :- 'Charles' frequents B.
'Camilla' frequents B :- B serves 'Youngs'.

% alternatively (equivalently)
% 'Camilla' frequents B :-
%     ('Charles' frequents B ; B serves 'Youngs').
```