Exercise 7

Foundations of Logic Programming

Prof. Horst Reichel, Sebastian Haufe

International Master Program in Computational Logic — winter term 2010/2011

Date of Exercise: 01.02.2011

Exercise 7.1 cf. Exercise 6.3

Consider the program P:

$$
\begin{array}{ll}
(1) & num(0) \\
(2) & num(s(x)) \leftarrow num(x) \\
(3) & odd(s(x)) \leftarrow \neg odd(x), num(x)
\end{array}
$$

e) Show that $odd(s(0))$ is not a semantic consequence of P.

Exercise 7.2 cf. Exercise 6.4

Take the following program P:

$$
\begin{array}{l}
p \leftarrow \\
p \leftarrow p \\
q \leftarrow r \\
q \leftarrow \neg r, p \\
r \leftarrow \neg p \\
t \leftarrow q \\
t \leftarrow r, \neg q
\end{array}
$$

a) Construct the dependency graph D_P of P.

b) Give a stratification of P.

c) Using your stratification show how to compute the standard model M_P of P.

Exercise 7.3

Consider the following program P:

$$
\begin{array}{l}
p(x) \leftarrow q(x), \neg p(b) \\
p(b) \leftarrow \neg q(b) \\
q(a) \leftarrow
\end{array}
$$

a) Give all Herbrand interpretations $I \subseteq HB_{\{p,q\},\{a,b\}}$ that are a model for P.

b) Which of the Herbrand models from a) are supported?

c) Give all standardized Herbrand interpretations $I_=$ that are a model for $comp(P)$.

Exercise 7.4

Consider the following program P:

$$
\begin{array}{l}
edge(a,b) \leftarrow \\
edge(b,c) \leftarrow \\
path(X,Y) \leftarrow edge(X,Y) \\
path(X,Y) \leftarrow edge(X,Z), path(Z,Y)
\end{array}
$$

a) Without using Lemma 6.8 and Corollary 6.9, show that P is not recurrent.

b) Give a level mapping $||$ and an interpretation I such that P is acceptable w.r.t. $||$ and I.

c) Is P still acceptable when adding the fact $edge(c,a) \leftarrow$ to it? Verify your claim by using Corollary 6.24 on Slide VIII/33.

Exercise 7.5

Download the meta interpreter for coinductive logic programming from the course website (there you will also find instructions on how to run it). Reconsider predicate $comember/2$ from the lecture:

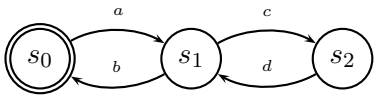
```
drop(H,[H|T],T).
drop(H,[_|T],T1) :- drop(H,T,T1).

:- coinductive comember/2.
comember(X,L) :-  drop(X,L,L1), comember(X,L1).
```

a) Define a coinductive predicate $stream/1$ which exactly accepts the infinite streams of natural numbers (represented by terms $0, s(0), s(s(0)), \ldots$) where each occurring 0 is directly succeeded by an even number and there are infinitely many such pairs.

Queries for testing:

`?- stream([0,s(0)]).`	Answer: `No`
`?- X = [0|X], stream(X).`	Answer: `Yes`
`?- X = [s(s(0))|X], stream([0|X]).`	<diverges>

b) Define a coinductive predicate $automaton/2$ such that $automaton(Str, S)$ succeeds iff the ω-automaton given in the Figure below accepts stream Str starting in state S. It only accepts infinite streams which pass the final state s_0 infinitely often.



Queries for testing:

`?- X = [d,b,a,b,a,c], automaton(X,s2).`	Answer: `No`
`?- X = [c,d,c,d,b,a|X], automaton(X,s1).`	Answer: `Yes`
`?- X = [c,d|X], automaton([a|X],s0).`	<diverges>