CSU44060 – Homework 2

1. The prove/2 predicate will fail to realise that `g` is a logical consequence of a Knowledge Base (KB) If KB contains a cyclical set of propositional clauses. An example of such KB would be:

   KB = [[g, a], [a, g], [a]]

   In this KB both `a` and `g` are logical consequences of the KB as: `a` is true for all models of KB as , and therefore `g` is true for all models of KB as (g ←a). However, the prove/2 predicate will fail to detect this as, due to the recursive nature of the predicate, it gets caught in the cyclical structure of the knowledge base and as a result, never reaches the base recursive statement of prove([], KB).

2. (a) See Below

```
/**
*   clause_satisfied/2
*      determines whether the clause selected from the KB is true based on an if-else
*      statement
*      - If the head of the rule is true, the clause is true
*      - Else if the body of the rule is true, this implies the head is true, the clause is
*        true
*      - Else the clause is not true
*   params
*      - H :- The head of the clause
*      - Body :- The body of the clause
*      - C :- The list of Logical Consequences of the current KB
*/
clause_satisfied([H|Body], C) :-
   \+ member(H, C),
   subset(Body, C).

/**
*   select_clause/4
*      selects a clause from the current KB and sends this clause to clause_satisfied/2
*      - If this clause is satisfied, it returns the head of that clause.
*   params
*      - KB :- The knowledge base being used currently
*      - C :- The list of logical consequences of this KB
*      - H :- The return value of the predicate. Is the head of a clause that is true.
*      - RestKB :- The remainder of the KB that has not been selected by this predicate
*/
select_clause(KB, C, H, RestKB) :-
   select(Clause, KB, RestKB),
   clause_satisfied(Clause, C),
    [H|_] = Clause.

/**
*   lc/3
*       Calls select_clause/4 to determine if there are any true clauses in the KB (or
*       that remaining of)
*       - If a true clause is found, the head of this clause (H returned from
*         select_clause/4 is appended
*         to the list of logical consequences and the predicate is called again with new
*         information).
*       - Else, the predicate returns the current list of logical consequences
*   params
*       KB :- The current KB in use
*       C :- The most up to date list of logical consequences
*       NewC :- The return list of logical consequences
*/
lc(KB, C, NewC) :-
    (select_clause(KB, C, H, RestKB) ->
        (append([H], C, UpdatedC),
        lc(RestKB, UpdatedC, NewC));
    NewC = C).

/**
*   lc/2
*       A predicate that when called and provided a KB returns a list of logical
*       consequences to that KB
*   params
*       KB :- The KB from which to determine all logical consequences
*       C :- The returned list of all logical consequences
*/
lc(KB, C) :- lc(KB, [], C).

/**
*   query/2
*       predicate provided in question that is used to determine whether X is a logical
*       consequence of KB
*       - Cut included at the end as each logical consequence only included once in C
*         and this prevents
*         further searching of the list if term is found.
*   params
*       X :- The term which is to be determined as a logical consequence of KB
*       KB :- The KB that may/may not have X as a logical consequence
*/
query(X, KB) :- lc(KB, C), member(X, C), !.
```

3. The V-H*erbrand-Interpretation* (IKKB) is a model of the knowledge base KB. This is due to the definition (iii) of IKKB *"for each n-ary predicate symbol p and n-tuple c1, … , cn from K, π(p)(c1, … , cn) = true ⇐⇒ p(c1, …, cn) is a logical consequence of KB."* If all *p(c1,…, cn)* are true in every model of KB, each *π(p)(c1,…, cn)* must evaluate as true, by definition. Therefore, as all clauses in IKKB are true in KB, IKKB is a model of KB

   Extending the bottom-up proof for Datalog to include V-Datalog is non-trivial as the current procedure relies on the fact that an atom remains true as more clauses are applied. With disjunctions, the truth of an atom can be invalidated by evaluating further clauses. However, as shown above the V-Herbrand-Interpretation (IKKB) is a model of KB. This information could be used to extend the bottom-up procedure for V-Datalog as, due to every clause in IKKB being true in every model of KB, when performing a bottom-up procedure, its soundness and completeness is maintained. Given a mechanical procedure, L, each logical consequence of KB will output by L, giving L completeness, and each output from L will be a logical consequence of KB, giving L soundness. This is due to only atoms that are a logical consequence of KB are included in its model IKKB.

4. $((\forall R)C) \sqcap (\exists R)C' \sqsubseteq (\exists R)C$

① Form counter example

$\neg((\exists R)C) \rightsquigarrow \forall R(\neg C)$

$\mathcal{A} := \{((\forall R)C \sqcap (\exists R)C')(a), (\forall R(\neg C))(a)\}$

② Negations pushed inside

③ Apply Rules

$\sqcap$-Rule on $(\forall R.C \sqcap \exists R.C')(a)$

$\mathcal{A} := \mathcal{A} \cup \{((\forall R)C)(a), ((\exists R)C')(a)\}$

$\exists$-Rule on $(\exists R.C')(a)$

$\mathcal{A} := \mathcal{A} \cup \{(R(a,b)), C'(b)\}$

$\forall$-Rule on $(\forall R.C)(a)$

$\mathcal{A} := \mathcal{A} \cup \{C(b)\}$

...

$\forall$-Rule on $(\forall R.\neg C)(a)$

$\mathcal{A} := \mathcal{A} \cup \{(\neg C)(b)\}$

... contradiction as $C(b) \sqcap \neg C(b)$

However

no contradiction to $C'(b)$

$\therefore$ subsumption does not hold

5. $\forall R(C \sqcup C') \sqsubseteq (\forall R)C \sqcup (\forall R)C'$

① Form counter Example

$\neg(\forall R(C) \sqcup \forall R(C'))$

$\rightsquigarrow \neg(\forall R(C)) \sqcap \neg(\forall R(C'))$

$\rightsquigarrow \exists R(\neg C) \sqcap \exists R(\neg C')$

$\mathcal{A} := \{ \forall R(C \sqcup C')(a), (\exists R(\neg C) \sqcap \exists R(\neg C'))(a) \}$

② Negations pushed inside

③ Apply Rules

$\sqcap$-Rule on $(\exists R(\neg C) \sqcap \exists R(\neg C'))(a)$

$\mathcal{A} := \mathcal{A} \cup \{ (\exists R(\neg C))(a), (\exists R(\neg C'))(a) \}$

$\sqcup$-Rule on $(\forall R(C \sqcup C'))(a)$

$\mathcal{A}_1 := \mathcal{A} \cup \{ (\forall R(C))(a) \}$

$\mathcal{A}_2 := \mathcal{A} \cup \{ (\forall R(C'))(a) \}$

$\exists$-Rule on $((\exists R(\neg C))(a)$ on $\mathcal{A}_1$

$\mathcal{A}_1 := \mathcal{A}_1 \cup \{ R(a,b), \neg C(b) \}$

$\forall$-Rule on $(\forall R(C))(a)$ on $\mathcal{A}_1$

$\mathcal{A}_1 := \mathcal{A}_1 \cup \{ C(b) \}$

... contradiction as $C(b) \sqcap \neg C(b)$

5. contd.

$\exists$ - Rule on [illegible] $(\exists R(\neg C'))(a)$ on A_2

$A_2 := A_2 \cup \{R(a,c), \neg C'(c)\}$

$\forall$ -Rule on $(\forall R(C'))(a)$ on A_2

$A_2 := A_2 \cup \{C'(c)\}$

... contradiction as $C'(c) \sqcap \neg C'(c)$

$\therefore$ subsumption holds.