Artificial Intelligence 1
Assignment8: First-Order Logic
– Given Dec 20, Due Jan 08 –

Problem 8.1 (Propositional Logic in Prolog) 25 pt

We implement propositional logic in Prolog.

We use the following Prolog terms to represent Prolog formulas

- lists of strings for signatures (each element being the name of a propositional variables)
- `var(s)` for a propositional variable named `s`, which is a string,
- `neg(F)` for negation,
- `disj(F,G)` for disjunction,
- `conj(F,G)` for conjunction,
- `impl(F,G)` for implication.

1. Implement a Prolog predicate `isForm(S,F)` that checks if `F` is well-formed formula relative to signature `isForm(S)`.

   Examples:

   ```
   ?- isForm(["a","b"],neg(var("a"))).
   True
   ```

   ```
   ?- isForm(["a","b"],neg(var("c"))).
   False
   ```

   ```
   ?- isForm(["a","b"],conj(var("a"),impl(var("b")))).
   False
   ```

2. Implement a Prolog predicate `simplify(F,G)` that replaces all disjunctions and implications with conjunction and negation.

   Examples:

   ```
   ?- simplify(disj(var("a"),var("b")), X).
   X = not(and(not(var("a")),not(var("b")))).
   ```

   Note that there is more than one possible simplification of a term, so your results may be different (but should be logically equivalent).

3. Implement a predicate `eval(P,F,V)` that evaluates a formula under assignment `P`. Here `P` is a list of terms `assign(s,v)` where `s` is the name of a propositional variable and `v` is a truth value (either 1 or 0). You can assume that `P` provides exactly one assignment for every propositional variable in `F`.

   Example:

```
?- eval([assign("a",1),assign("b",0)], conj(var("a"), var("b")), V).
V = 0.

?- eval([assign("a",1),assign("b",1)], conj(var("a"), var("b")), V).
V = 1.
```

Solution:

```
contains([H|_],H).
contains([_|L],X) :- contains(L,X).

% isForm(S,F) holds if F is a PL-formula over signature S
% the signature is given as a list of names of propositional variables
isForm(S,var(N)) :- string(N), contains(S,N).
isForm(S,neg(F)) :- isForm(S,F).
isForm(S,conj(F,G)) :- isForm(S,F), isForm(S,G).
isForm(S,disj(F,G)) :- isForm(S,F), isForm(S,G).
isForm(S,impl(F,G)) :- isForm(S,F), isForm(S,G).

% simplify(F,G) holds if G is the result of replacing in F
% disjunction and implication with conjunction and negation
simplify(var(S),var(S)).
simplify(neg(F), neg(FS)) :- simplify(F,FS).
simplify(conj(F,G), conj(FS,GS)) :- simplify(F,FS), simplify(G,GS).
simplify(disj(F,G), neg(conj(neg(FS),neg(GS)))) :- simplify(F,FS), simplify(G,GS).
simplify(impl(F,G), neg(conj(FS,neg(GS)))) :- simplify(F,FS), simplify(G,GS).

% eval(P,F,V) holds if I_P(F) = 1
% the assignment P is given as a list [assign(N,V), ...]
% where N is the name of a propositional variable and V is 0 or 1
eval(P,var(N), V) :- contains(P,assign(N,V)).
eval(P,neg(F), V) :- eval(P,F,FV), V is 1-FV.
eval(P,conj(F,G), V) :- eval(P,F,FV), eval(P,G,GV), V is FV*GV.
eval(P,disj(F,G), V) :- eval(P,F,FV), eval(P,G,GV), V is FV+GV-FV*GV.
eval(P,impl(F,G), V) :- eval(P,F,FV), eval(P,G,GV), V is (1-FV)+GV-(1-FV)*GV.
```

Problem 8.2 (PL Semantics) 20 pt

We work with a propositional logic signature declaring variables A and B and consider the following two formulas:

1. $A \Rightarrow (B \Rightarrow A)$
2. $(A \wedge B) \Rightarrow (A \wedge C)$

We use a fixed but arbitrary assignment φ for the propositional variables.

For each of the two formulas F, apply the definition of the interpretation $\mathcal{I}_\varphi(F)$ step-by-step to obtain the semantic condition that F holds under φ. Afterwards determine if F is valid or not by one of the following:

- argue why $\mathcal{I}_\varphi(F)$ is true, which means F is valid because it holds for an arbitrary φ,

- give an assignment φ that makes $\mathcal{I}_\varphi(F)$ false

---

Solution: We use $\top/\bot$ as the two truth values here. They are sometimes also written as $1/0$ or T/F.

- $A \Rightarrow (B \Rightarrow A)$ is valid:

  For any assignment φ:

$$\begin{aligned}\mathcal{I}_\varphi(A \Rightarrow (B \Rightarrow A)) = \mathcal{I}_\varphi(\neg(A \wedge \neg\neg(B \wedge \neg A))) \\ = \top \text{ iff } \mathcal{I}_\varphi(A \wedge \neg\neg(B \wedge \neg A)) = \bot \\ \text{iff not both } \varphi(A) = \top \text{ and } \mathcal{I}_\varphi(\neg\neg(B \wedge \neg A)) = \top \\ \text{The latter is the case iff } \mathcal{I}_\varphi(B \wedge \neg A) = \top \\ \text{iff } \varphi(B) = \top \text{ and } \varphi(A) = \bot\end{aligned}$$

  So for the formula is false iff both $\mathcal{I}_\varphi(A) = \top$ and $\mathcal{I}_\varphi(A) = \bot$.

- $(A \wedge B) \Rightarrow (A \wedge C)$: Not valid. Counterexample: $\varphi(A) = \varphi(B) = \top$, $\varphi(C) = \bot$.

Problem 8.3 (FOL-Signatures) 20 pt

1. Model the following situation as a FOL signature. (FOL and PLNQ signatures are the same.)
   - We have constants (= nullary functions) called `zero` and `one`.
   - We have a binary function called `plus`.
   - We have a unary function called `minus`.
   - We have a binary predicate called `less`.
2. Now consider the signature given by
   - $\Sigma_0^f = \{a, b\}$
   - $\Sigma_1^f = \{f, g\}$
   - $\Sigma_2^f = \{h\}$
   - $\Sigma_0^p = \{p\}$
   - $\Sigma_1^p = \{q\}$
   - $\Sigma_2^p = \{r\}$
   - all other sets empty

   Give

- a term over this signature that uses all function symbols
- a formula over this signature that uses all function and predicate symbols

---

Solution:

1. $\Sigma_0^f = \{\texttt{zero}, \texttt{one}\}$, $\Sigma_1^f = \{\texttt{minus}\}$, $\Sigma_2^f = \{\texttt{plus}\}$, $\Sigma_2^p = \{\texttt{less}\}$, and all other sets are empty
2. E.g., $t = h(f(a), g(b))$ for the term $r(t, t) \wedge q(t) \wedge p$ for the formula

---