Assignment6 – Propositional Logic

Problem 6.1 (PL Concepts)

Which of the following statements are true? In each case, give an informal argument why it is true or a counter-example.

1. Every satisfiable formula is valid.

Solution: Not true. Counter-example: p is satisfiable (put $\varphi(p) = \mathtt{T}$) but not valid (falsified by $\varphi(p) = \mathtt{F}$).

---

2. Every valid formula is satisfiable.

Solution: True. Assume F is valid. Then F is satisfied by all assignments. We know (This is a subtle step that can easily be overlooked.) that there is at least one assignment a. (Even if there are no propositional variables, we would still have the empty assignment.) So a must satisfy F and therefore F is satisfiable.

---

3. If A is satisfiable, then $\neg A$ is unsatisfiable.

Solution: Not true. Counter-example: p is satisfiable (put $\varphi(p) = \mathtt{T}$), but $\neg p$ is also satisfiable (put $\varphi(p) = \mathtt{F}$).

---

4. If $A \models B$, then $A \wedge C \models B \wedge C$.

Solution: True. Assume $A \models B$ (H) and an assignment φ such that $I_\varphi A \wedge C = \mathtt{T}$ (A). We need to show that also $I_\varphi A \wedge C = \mathtt{T}$ (G).
By definition, (A) yields $I_\varphi(A) = \mathtt{T}$ (A1) and $I_\varphi(C) = \mathtt{T}$ (A2).
By definition of (H), we obtain from (A1) that $I_\varphi(B) = \mathtt{T}$ (B).
Then we obtain (G) from its definition and (B) and (A2).

---

5. Every admissible inference rule is derivable.

Solution: Not true. Counter-example: The empty derivation relation has no inference rules and thus no derivable formulas. Then any rule with non-empty set of assumptions is admissible. But no rule is derivable.

---

6. If $\vdash$ is sound for $\vDash$ and $\{A, B\} \vdash C$, then C is satisfiable if A and B are.

Solution: Not true. The assumptions do show that $A, B \vDash C$. So if we have an assignment that satisfies both A and B, then that assignment also satisfies C and thus C is satisfiable. But we only know that A and B are satisfiable by some assignments, not necessarily the same one. A counter-example, is $A = p, B = \neg p, C$ any unsatisfiable formula. Then $A, B \vDash C$ holds (because there are no assignments that satisfy both A and B), and A and B but not C are satisfiable.

Problem 6.2 (Propositional Logic in Prolog)

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 `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 = neg(conj(neg(var("a")),neg(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) = V
% 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 6.3 (PL Semantics)

We work with a propositional logic signature declaring variables A and B. For each of the fomulae blow 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

1. $A \Rightarrow (B \Rightarrow A)$

Solution: $A \Rightarrow (B \Rightarrow A)$ is valid: For any assignment φ:

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

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

---

2. $A \wedge B \Rightarrow A \wedge C$

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