School of Software Engineering, USTC (Suzhou)
Middle Term Exam Paper for Academic Year 2020-2021-1
Open or Close: Open

Course: Formal Methods _____ Time: Nov. 27th, 2020 _____
Student Name: _____ Student No._____
Class: _____ Score:_____

I: Propositional Logic

Given the following inference rules for propositional logic:

$$\frac{}{\Gamma, P \vdash P}\ (Var) \qquad \frac{}{\Gamma \vdash T}\ (\top)$$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash P}\ (\bot E) \qquad \frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash P \wedge Q}\ (\wedge I)$$

$$\frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash P}\ (\wedge E_1) \qquad \frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash Q}\ (\wedge E_2)$$

$$\frac{\Gamma \vdash P}{\Gamma \vdash P \vee Q}\ (\vee I_1) \qquad \frac{\Gamma \vdash Q}{\Gamma \vdash P \vee Q}\ (\vee I_2)$$

$$\frac{\Gamma, P \vdash Q}{\Gamma \vdash P \to Q}\ (\to I) \qquad \frac{\Gamma \vdash P \to Q \quad \Gamma \vdash P}{\Gamma \vdash Q}\ (\to E)$$

$$\frac{\Gamma, P \vdash \bot}{\Gamma \vdash \neg P}\ (\neg I) \qquad \frac{\Gamma \vdash P \quad \Gamma \vdash \neg P}{\Gamma \vdash \bot}\ (\neg E)$$

$$\frac{\Gamma \vdash P \vee Q \quad \Gamma, P \vdash R \quad \Gamma, Q \vdash R}{\Gamma \vdash R}\ (\vee E) \qquad \frac{\Gamma \vdash \neg\neg P}{\Gamma \vdash P}\ (\neg\neg E)$$

1. **[6 points]** Prove the validity of the following judgment, by drawing the proof tree:

$$(P \wedge Q) \vdash P \to Q$$

2. **[6 points]** Prove the validity of the following judgment, by drawing the proof tree:

$$\vdash (P \vee Q) \to (\neg Q \to P)$$

II: Constructive Logic

3. **[8 points]** Try to prove that there must exist two irrational numbers p and q, such that

$$p^q$$

is rational. Is your proof constructive or not? Please give your reason.

III: SAT

Here are the rules for eliminating implication:

$$
\begin{aligned}
C(p) &= p \\
C(\neg P) &= \neg C(P) \\
C(P \wedge Q) &= C(P) \wedge C(Q) \\
C(P \vee Q) &= C(P) \vee C(Q) \\
C(P \rightarrow Q) &= \neg C(P) \vee C(Q)
\end{aligned}
$$

Rules for conversion into NNF (Negation Normal Form):

$$
\begin{aligned}
C(p) &= p \\
C(\neg p) &= \neg p \\
C(\neg\neg P) &= C(P) \\
C(P \wedge Q) &= C(P) \wedge C(Q) \\
C(P \vee Q) &= C(P) \vee C(Q) \\
C(\neg(P \wedge Q)) &= C(\neg P) \vee C(\neg Q) \\
C(\neg(P \vee Q)) &= C(\neg P) \wedge C(\neg Q)
\end{aligned}
$$

Rules for converting into CNF (Conjunction Normal Form):

$$
\begin{aligned}
C(p) &= p \\
C(\neg p) &= \neg p \\
C(P \wedge Q) &= C(P) \wedge C(Q) \\
C(P \vee Q) &= D(C(P), C(Q)) \\
D(P = P_1 \wedge P_2, Q) &= D(P_1, Q) \wedge D(P_2, Q) \\
D(P, Q = Q_1 \wedge Q_2) &= D(P, Q_1) \wedge D(P, Q_2) \\
D(P, Q) &= P \vee Q
\end{aligned}
$$

4. [12 points] Suppose we have proposition F as following:

$$\neg(p \rightarrow (q \wedge (\neg p \rightarrow q)))$$

Questions:

1. Please eliminate implications in the proposition F, by using the above rules.
2. Please convert your answer in question 1 to NNF, by using the above rules.
3. Please convert your answer in question 2 to CNF, by using the above rules.

5. [12 points] Some modern satisfiability engines for propositional logic are based on the Davis-Putnam-Logemann-Loveland algorithm (DPLL), which decides the satisfiability of propositions in CNF. The basic implementation of DPLL can described by the following pseudo-code:

```
1  DPLL(F){
2    newF = BCP(F)
```

```
3   if(newF == ⊤)
4     return sat;
5   if(newF == ⊥)
6     return unsat;
7
8   x = select_var(newF);
9   if(DPLL(newF[x ↦ ⊤]))
10     return sat;
11  return DPLL(newF[x ↦ ⊥])
12 }
```

The BCP() method at line 2 stands for Boolean Constraint Propagation, which is based on unit resolution. Unit resolution deals with one unit clause, which must be p or $\neg p$, and one clause contains the negation of the unit clause. Then unit resolution is the deduction:

$$\frac{p \qquad C(\neg p)}{C[\bot]} \qquad \text{(Unit-Resol)}$$

In line 2 of the DPLL algorithm, the BCP() function will apply the above unit resolution as much as possible.

Suppose we call the function DPLL() with the following proposition F:

$$(\neg p_1 \vee p_3) \wedge (\neg p_2 \vee p_3 \vee p_4) \wedge (p_1 \vee \neg p_3 \vee \neg p_4) \wedge (p_1)$$

Questions:

1. For the first recursive call, on line 2 of DPLL(), what's the value for $newF$? Please write down your answer.
2. For the first recursive call, which variable you'll choose at line 8 of the DPLL() function? Please give reasons for your choice.
3. What's the final result for the function DPLL()? Is the proposition F satisfiable or not?

6. [10 points] Alice, Bob and Carol will take 3 seats as Figure shows. And there are some constraints:

Figure 1: seat arrangement

- Alice cannot sit near to Carol;

- Bob can not sit right to Alice;
- Everyone should take one and only one seat;
- Every seat is just taken by one person.

we use the propositions

$$A_i, \text{where } 1 \leq i \leq 3,$$

to describe the situation that Alice takes the i-th seat, for instance, the proposition A_1 denotes that Alice will take the seat 1, and so on. Similarly, we use the propositions

$$B_i, \text{where } 1 \leq i \leq 3,$$

and

$$C_i, \text{where } 1 \leq i \leq 3,$$

to describe that Bob or Carol will take some seat. Please write down the logic proposition to describe the constraints above. (You just need to write down the propositions, you don't need to simplify it, or to judge its satisfiability.)

IV: Predicate Logic

Here are the inference rules for predicate logic (other rules are same with propositional logic in Section I):

$$\frac{\Gamma, x \vdash P}{\Gamma \vdash \forall x.P}\ (\forall I) \qquad \frac{\Gamma \vdash \forall x.P}{\Gamma \vdash P[x \mapsto E]}\ (\forall E)$$

$$\frac{\Gamma \vdash P[x \mapsto E]}{\Gamma \vdash \exists x.P}\ (\exists I) \qquad \frac{\Gamma \vdash \exists x.P \quad \Gamma, x, P \vdash Q}{\Gamma \vdash Q}\ (\exists E)$$

7. [6 points] Please prove the validity of the following proposition, by drawing the proof tree :

$$\forall x.(P(x) \to Q(x)) \vdash \forall x.(P(x)) \to \forall x.(Q(x))$$

Here are the substitution rules for predicate logic:

$$\begin{aligned}
x[x \mapsto E] &= F \\
y[x \mapsto E] &= y, \text{where } x \neq y \\
f(E_1, ..., E_n)[x \mapsto E] &= f(E_1[x \mapsto E], ..., E_n[x \mapsto E]) \\
r(E_1, ..., E_n)[x \mapsto E] &= r(E_1[x \mapsto E], ..., E_n[x \mapsto E]) \\
(P_1 \wedge P_2)[x \mapsto E] &= P_1[x \mapsto E] \wedge P_2[x \mapsto E] \\
(P_1 \vee P_2)[x \mapsto E] &= P_1[x \mapsto E] \vee P_2[x \mapsto E] \\
(P_1 \to P_2)[x \mapsto E] &= P_1[x \mapsto E] \to P_2[x \mapsto E] \\
(\forall x.P)[x \mapsto E] &= \forall x.P \\
(\forall y.P)[x \mapsto E] &= (\forall z.P[y \mapsto z])[x \mapsto E], \text{where } z \text{ is fresh} \\
(\exists x.P)[x \mapsto E] &= \exists x.P \\
(\exists y.P)[x \mapsto E] &= (\exists z.P[y \mapsto z])[x \mapsto E], \text{where } z \text{ is fresh}
\end{aligned}$$

8. [10 points] Given the following proposition F

$$\exists x.(P(y,x) \wedge \forall y.(\neg Q(y,x)) \vee P(y,z)),$$

where both P and Q are predicate symbols with two arguments. Questions:

1. Please write down both the bound and free variable sets for the proposition F.
2. Please write down the result of the substitution

$$F[y \mapsto x].$$

3. Suppose R is a predicate symbols with two arguments, please write down result of the substitution

$$F[x \mapsto R(y,z)].$$

V: Theory for EUF

9. [10 points] The Union-Find (UF) algorithm is very efficient to solve EUF problems. Suppose we have the following equalities

$$a = c, b = d, a = e, d = h, f = g, g = e$$

Questions:

1. If we use the Union-Find algorithm to solve these equalities, what does the data structure look like? Please draw some sketches. (You don't need to write down the algorithm details.)
2. Suppose we add another equality

$$e = d$$

into this group, what does the data structure look like, after we use union-find algorithm to solve these equalities?

10. [10 points] One important application of the EUF theory is proving program equivalence. In the following, we present two implementations of the same algorithm, one is:

```
1 int calculate_a(int in_1, int in_2){
2   int out_a_1 = in_1 * in_2;
3   int out_a_2 = in_1 + in_2;
4   int out_a = out_a_1 - out_a_2;
5   return out_a;
6 }
```

and the other one is:

```
1 int calculate_b(int in_1, int in_2){
2   int out_b = (in_1 * in_2) - (in_1 + in_2);
3   return out_b;
4 }
```

Questions:

1. Please describe the basic idea to prove these two algorithms are equivalent, by using EUF theory. Please write down the logical proposition F you need to prove.

2. Bob wants to prove the above proposition F, by using the Z3 solver, the code he wrote looks like:

```
1  solver = Solver()
2  solver.add(F)
3  print(solver.check())
```

   Bob ran Z3 on this code, and got the result `sat`. Did Bob successfully prove that two algorithms are equivalent? If yes, please give your reason; if not, please write down the correct code.

VI: Formal Methods Foundation

We'd like to hear your opinions about this course, so please answer the following questions. (Any answer, except no answer, will receive FULL credit.)

1. **[3 points]** What is the best aspect of this course?

2. **[3 points]** What is the worst aspect of this course?

3. **[4 points]** What we can do to make this course better?

End of Test.