School of Software Engineering, USTC (Suzhou)
Final Term Exam Paper for Academic Year 2022-2023-1
Open or Close: Open

Course: Formal Methods ____________ Time: Feb 20th, 2023 ____________

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:

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

II: Constructive Logic

2. [**6 points**]Given the following proposition:

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

Does this proposition hold in constructive logic? Explain your conclusion.

III: SAT

Here are the rules for eliminating implication:

$$
\begin{aligned}
\mathcal{E}(\top) &= \top \\
\mathcal{E}(\bot) &= \bot \\
\mathcal{E}(p) &= p \\
\mathcal{E}(P \wedge Q) &= \mathcal{E}(P) \wedge \mathcal{E}(Q) \\
\mathcal{E}(P \vee Q) &= \mathcal{E}(P) \vee \mathcal{E}(Q) \\
\mathcal{E}(P \to Q) &= \mathcal{E}(\neg P) \vee \mathcal{E}(Q) \\
\mathcal{E}(\neg P) &= \neg \mathcal{E}(P)
\end{aligned}
$$

Rules for conversion into NNF (Negation Normal Form):

$$
\begin{aligned}
\mathcal{N}(\top) &= \top \\
\mathcal{N}(\bot) &= \bot \\
\mathcal{N}(p) &= p \\
\mathcal{N}(\neg P) &= \neg \mathcal{N}(P) \\
\mathcal{N}(\neg\neg P) &= \mathcal{N}(P) \\
\mathcal{N}(P \wedge Q) &= \mathcal{N}(P) \wedge \mathcal{N}(Q) \\
\mathcal{N}(P \vee Q) &= \mathcal{N}(P) \vee \mathcal{N}(Q) \\
\mathcal{N}(\neg(P \wedge Q)) &= \mathcal{N}(\neg P) \vee \mathcal{N}(\neg Q) \\
\mathcal{N}(\neg(P \vee Q)) &= \mathcal{N}(\neg P) \wedge \mathcal{N}(\neg Q)
\end{aligned}
$$

Rules for converting into CNF (Conjunction Normal Form):

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

3. [10 points] Suppose we have proposition F as following:

$$\neg((p \to q) \wedge (q \to r)) \wedge (p \to r)$$

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.

4. [**6 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     if(DPLL(newF[x ↦ ⊤]))
9       return sat;
10   return DPLL(newF[x ↦ ⊥])
11 }
```

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:

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

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

Describe the execution of DPLL on the following formula:

$$(p \vee q \vee r) \wedge (\neg p \vee q) \wedge (p \vee \neg q) \wedge (p \vee r) \wedge (\neg p \vee \neg r)$$

IV: Predicate Logic

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

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

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

5. [**8 points**] Please prove the validity of the following proposition, by drawing the proof tree :

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

Here are the substitution rules for predicate logic:

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

6. [7 points] Given the following proposition F

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

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

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

V: Theory for EUF

7. [8 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:

```c
int fun_1(int data[]){
  int i, out_a;
  out_a = *data;
  for (i = 0; i < 2; i++){
    data = data + 1;
    out_a = out_a + *data;
  }
  return out_a;
}
```

and the other one is:

```c
int fun_2(int data[]){
  int out_b = 0;
  out_b = *data + *(data + 1) + *((data + 1) + 1);
  return out_b;
}
```

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:

```
solver = Solver()
solver.add(F)
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: Linear Arithmetic

8. [**10 points**] Given the following linear inequalities with five variables x, y, z, m, n and eight constraints:

$$\begin{cases} 3x - 2y + n \geq 1 \\ x + y + m = 3 \\ y + 4m - n \leq 4 \\ 3m - 2n + z \geq 5 \\ 4x - 3m \leq 1 \\ x \leq 1 \\ y \geq -2 \\ n \leq 1 \end{cases}$$

Question: Use Fourier-Motzkin variable elimination algorithm to determine whether the liner inequalities can be satisfied, please write down the detailed procedures. And if it is satisfied, give a solution.

VII: Theories for Data Structures

9. [**9 points**] The most commonly used decision procedure for bit-vector arithmetic is called $bit-blasting$. The following algorithm implements this technique. For a given bit-vector arithmetic formula ϕ , the algorithm computes an equisatisfiable propositional formula β ,which is then passed to a SAT solver.

```
bitBlast(P){
  // convert the proposition to atomic bools
  blastProp(P);
  // generate constraints
  genConsProp(P);
}
```

Questions: Assuming that x, y, and z are all 2-bit bit vectors, please write the final constraint logic expression of the formula $(x \neq 2) \wedge (y\&2 = z) \wedge (x|2 = 3)$ according to the bit-blasting algorithm.

10. [10 points] Logic with pointers can be converted into EUF problem, by eliminating pointers through encoding their semantics using the store function S and heap function H. To simplify things, we assume that the heap only contains values of integer type, and the address is also of integer type:

$$S : \texttt{int} \to \texttt{int}$$
$$H : \texttt{int} \to \texttt{int}$$

The rules to eliminate a pointer T are:

$$
\begin{aligned}
[\![x]\!] &= H(S(x)) \\
[\![T + E]\!] &= [\![T]\!] + [\![E]\!] \\
[\![\&x]\!] &= S(x) \\
[\![\& * T]\!] &= [\![T]\!] \\
[\![*T]\!] &= H([\![T]\!]) \\
[\![\texttt{NULL}]\!] &= 0
\end{aligned}
$$

The rules to eliminate an expression E are:

$$
\begin{aligned}
[\![n]\!] &= n \\
[\![x]\!] &= H(S(x)) \\
[\![E + E]\!] &= [\![E]\!] + [\![E]\!] \\
[\![E - E]\!] &= [\![E]\!] - [\![E]\!] \\
[\![*T]\!] &= H([\![T]\!])
\end{aligned}
$$

The rules to eliminate a relation R are:

$$
\begin{aligned}
[\![E = E]\!] &= [\![E]\!] = [\![E]\!] \\
[\![E \neq E]\!] &= [\![E]\!] \neq [\![E]\!] \\
[\![E < E]\!] &= [\![E]\!] < [\![E]\!] \\
[\![T = T]\!] &= [\![T]\!] = [\![T]\!] \\
[\![T \neq T]\!] &= [\![T]\!] \neq [\![T]\!] \\
[\![T < T]\!] &= [\![T]\!] < [\![T]\!]
\end{aligned}
$$

The rules to eliminate a proposition P are:

$$
\begin{aligned}
[\![P \to Q]\!] &= [\![P]\!] \to [\![Q]\!] \\
[\![P \wedge Q]\!] &= [\![P]\!] \wedge [\![Q]\!] \\
[\![\neg R]\!] &= \neg [\![R]\!]
\end{aligned}
$$

Questions:

1. Translate the proposition F

$$(*q = \&a) \wedge (a[1] = 1) \to *(*q + 1) = 1.$$

with pointers to EUF theory proposition $F1$ by using the above rules.

2. If we extend our memory model by introducing a new sort of memory

$$R : \texttt{int} \to \texttt{int}$$

which store pure variables, and we can introduce a new elimination rule

$$[\![x]\!] = R(x).$$

Try to translate the proposition F to EUF theory proposition F2 with the extended memory model.

3. If we assume that the heap contains two type of values: address type and integer type, then we will have two heap function:

$$H_a : \texttt{addr} \to \texttt{addr}$$
$$H_i : \texttt{addr} \to \texttt{int}$$

where the function H_a can get address type value from heap and the function H_i can get intger type value from heap. Try to translate the proposition F to EUF proposition F3 with the extended memory model.

VIII: Theory Combination

11. [10 points] Consider the formula F:

$$x_1 \geq 0 \land x_1 \leq 2 \land f(x_3) = f(x_1) \land (x_2 = 2 \lor x_2 = 0) \land x_1 = \mathit{store}\,[A, x_0, 0]\,[x_0] \land f(x_1) - f(x_2) = x_3$$

Question: Use DPLL(T) algorithm to decide sat of the formula F, write down the steps and result.

IX: Symbolic Execution

12. [10 points] Given the function f in C language :

```c
int f(int x, int y, int z) {
  while (x <= 2){
  x++;
  }
  int m = x*x*x;
  if (m == y){
   x = y/z;
  }
  return x;
}
```

Questions: Suppose we do the concolic execution on the function f to trigger the error. Write down the memory and path conditions the concolic execution engine will generate with the initial input:

$$x = -2,\ y = 2,\ z = 2$$