CMSC 433
Programming Language Technologies and Paradigms

Final Review

Questions

Question	Points
P1. Basic Concepts	10
P2. Dafny	25
P3. Hoare Logic	13
P4. SAT	15
P5. Symbolic Execution	16
P6. Rocq	21
Total	100

1. Basic Concepts

- True/False, Multiple-Choice Questions
  - Dafny
  - Floyd/Hoare Logic
  - SAT/SMT
  - Symbolic Execution
  - Rocq

Example 1

- Which of the following would cause a **termination error** in Dafny?

a) Missing `decreases` clause in a recursive function
b) Function without an `ensures` clause
c) Assertion that always evaluates to true
d) Loop with a valid invariant

Example 2

- What does the ensures clause in Dafny specify?

a) Preconditions that must hold before execution
b) Postconditions that must hold after execution
c) Loop invariants
d) Assertions checked at runtime

Example 3

- ghost variables in Dafny exist only for verification purposes and are not part of the compiled executable.

Dafny

- Mostly fill in the blanks
  - Concepts
  - ensures, requires, assert, assume
  - Function, method, lemma, predicate
  - Difference between function and method
  - Loop invariants

Fill in the Blanks

Complete the missing invariant to make the loop verifiable:

```
method CountDown(n: nat)
  ensures true
{
  var i := n;
  while i > 0
    invariant [                    ]
  {
    i := i - 1;
  }
}
```

Fill in the Blanks

Complete the missing invariant to make the loop verifiable:

```
method CountDown(n: nat) returns (x:int)
ensures x == 100 - 2 * n;
{
  var i := n;
  x := 100;
  while i > 0
    invariant [                          ]
  {
     i := i - 1;
     x := x - 2;
   }
}
```

Fill in the Blanks

Complete the missing invariant to make the loop verifiable:

```
method CountDown(n: nat) returns (x:int)
ensures x == 100 - 2 * n;
{
  var i := n;
  x := 100;
  while i > 0
    invariant x == 100 - 2 * (n-i)
  {
     i := i - 1;
     x := x - 2;
   }
}
```

Floyd Hoare Logic

- Hoare Triples
  - Assignment
  - Skip
  - Sequence
  - Conditional
  - While

Example

[ ]

if x ≤ 0

  { y := 2 }

else

  { y := x + 1 }

{ x ≤ y }

SAT & Z3

- CNF Formulas
- Converting a given formula to CNF
  - ~~Tseitin Transformation or~~ De Morgan's laws
- DPLL: Unit propagation, Pure Literals
- Z3 programming: like chicken & rabbit problem

Unit Propagation

- Given the CNF formula:

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

- a) Apply **unit propagation** step-by-step.
  b) Is the formula satisfiable? Justify.

Unit Propagation

- Given the CNF formula:

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

- a) Apply **unit propagation** step-by-step.
  b) Is the formula satisfiable? Justify.
- Solution:
  - Clause (p) → assign p = true.
  - Substitute in formula:
    - (¬p ∨ q) → (false ∨ q) → q = true.
  - Clause (¬q ∨ r) → (false ∨ r) → r = true.
  - But (¬r) → requires r = false.
  - Contradiction. Unsatisfiable.

Example

- Explain what a **pure literal** is and how the DPLL algorithm uses it.

Example

- Explain what a **pure literal** is and how the DPLL algorithm uses it.

- A literal that appears with only one polarity (only positive or only negated) in all clauses.
- DPLL sets pure literals to make all clauses containing them true, simplifying the formula.
- If we have $(p \vee q) \wedge (\neg q \vee r) \wedge (p \vee r)$,
  $\rightarrow$ p and r are **pure** (only positive occurrences).

Symbolic Execution

```
f5(a,b){
  x = 1;
  y = 2;
  if(a != 10){
    y = x + 3;
    if(b == 0){
      x = 2 * (a + b);
    }
  }
  return x + y;
}
```

1:Conditions: [a == 10]
SAT Input: [a = 10]
x = 1, y = 0
2:Conditions: [a != 10, b == 0]
SAT Input: [a = 11, b = 0]
y = x + 3
x = 2 * (a + b)
3:Conditions: [a != 10, b != 0]
SAT Input: [b = 1, a = 11]
y = x + 3

Rocq

- Multiple Choice
- Fill in the blanks
- Simple proofs
- Tactics:
  - simpl, intros, induction, destruct, apply, rewrite, left/right, split,
  - assumption, contradiction, discriminate, reflexivity, trivial

Example:

```
Theorem plus_0_r : forall n : nat,
    n + 0 = n.
Proof.
```