CMSC 433
Programming Language Technologies and Paradigms

Midterm Review

Midterm Questions

Question	Points
P1. Basic Concepts	12
P2. Dafny	50
P3. Hoare Logic	10
P4. SAT	28
Total	100

1. Basic Concepts

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

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

- Which step of the DPLL algorithm assigns a variable value based on a clause with only one literal?

a) Backtracking
b) Pure literal elimination
c) Unit propagation
d) Clause learning

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: similar to 8-queens and sudoku

Tseitin Transformation

- F = (p ∨ (q∧r))
- x1↔ (q ∧ r)
- x2↔ (p ∨ x1)

- x1→(q∧r) becomes $(\neg x_1 \vee q) \wedge (\neg x_1 \vee r)$
- (q∧r)→x1 becomes $(\neg q \vee \neg r \vee x_1)$

- x2→(p∨x1) becomes $(\neg x_2 \vee p \vee x_1)$
- (p∨x1)→x2 becomes $(\neg p \vee x_2) \wedge (\neg x_1 \vee x_2)$

- Final CNF: $(\neg x_1 \vee q) \wedge (\neg x_1 \vee r) \wedge (\neg q \vee \neg r \vee x_1) \wedge (\neg x_2 \vee p \vee x_1) \wedge (\neg p \vee x_2) \wedge (\neg x_1 \vee x_2)$

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).