ECE 653, Assignment 2

Weiqiang Yu
w8yu@uwaterloo.ca
21020022

November 2023

1 Q1 Solution

(a) There are four execution paths in Prog1. They are:

- 1, 2, 3, 4, 9, 11, 12, 13, 17
- 1, 2, 5, 6, 7, 9, 11, 12, 13, 17
- 1, 2, 3, 4, 9, 11, 14, 15, 16, 17
- 1, 2, 5, 6, 7, 9, 11, 14, 15, 16, 17

(b) The symbolic execution for each path and the resulting path conditions are shown in Table 1, 2, 3 and 4.

Table 1: Symbolic Execution of Path1

Edge	Symbolic State (PV)	Path Condition (PC)
$1 \rightarrow 2$	x $\rightarrow X_0$, y $\rightarrow Y_0$	true
$2 \rightarrow 3$	x $\rightarrow X_0$, y $\rightarrow Y_0$	$X_0 + Y_0 > 15$
$3 \rightarrow 4$	x $\rightarrow X_0 + 7$, y $\rightarrow Y_0$	$X_0 + Y_0 > 15$
$4 \rightarrow 9$	x $\rightarrow X_0 + 7$, y $\rightarrow Y_0 - 12$	$X_0 + Y_0 > 15$
$9 \rightarrow 11$	x $\rightarrow X_0 + 9$, y $\rightarrow Y_0 - 12$	$X_0 + Y_0 > 15$
$11 \rightarrow 12$	x $\rightarrow X_0 + 9$, y $\rightarrow Y_0 - 12$	$X_0 + Y_0 > 15 \wedge 2 * (X_0 + Y_0 - 3) > 21$
$12 \rightarrow 13$	x $\rightarrow 3 * (X_0 + 9)$, y $\rightarrow Y_0 - 12$	$X_0 + Y_0 > 15 \wedge 2 * (X_0 + Y_0 - 3) > 21$
$13 \rightarrow 17$	x $\rightarrow 3 * (X_0 + 9)$, y $\rightarrow 2 * (Y_0 - 12)$	$X_0 + Y_0 > 15 \wedge 2 * (X_0 + Y_0 - 3) > 21$

Table 2: Symbolic Execution of Path2

Edge	Symbolic State (PV)	Path Condition (PC)
$1 \to 2$	x $\to X_0$, y $\to Y_0$	true
$2 \to 5$	x $\to X_0$, y $\to Y_0$	$X_0 + Y_0 \leq 15$
$5 \to 6$	x $\to X_0$, y $\to Y_0$	$X_0 + Y_0 \leq 15$
$6 \to 7$	x $\to X_0$, y $\to Y_0 + 10$	$X_0 + Y_0 \leq 15$
$7 \to 9$	x $\to X_0 - 2$, y $\to Y_0 + 10$	$X_0 + Y_0 \leq 15$
$9 \to 11$	x $\to X_0$, y $\to Y_0 + 10$	$X_0 + Y_0 \leq 15$
$11 \to 12$	x $\to X_0$, y $\to Y_0 + 10$	$X_0 + Y_0 \leq 15 \wedge 2 * (X_0 + Y_0 + 10) > 21$
$12 \to 13$	x $\to 3 * X_0$, y $\to Y_0 + 10$	$X_0 + Y_0 \leq 15 \wedge 2 * (X_0 + Y_0 + 10) > 21$
$13 \to 17$	x $\to 3 * X_0$, y $\to 2 * (Y_0 + 10)$	$X_0 + Y_0 \leq 15 \wedge 2 * (X_0 + Y_0 + 10) > 21$

Table 3: Symbolic Execution of Path3

Edge	Symbolic State (PV)	Path Condition (PC)
$1 \to 2$	x $\to X_0$, y $\to Y_0$	true
$2 \to 3$	x $\to X_0$, y $\to Y_0$	$X_0 + Y_0 > 15$
$3 \to 4$	x $\to X_0 + 7$, y $\to Y_0$	$X_0 + Y_0 > 15$
$4 \to 9$	x $\to X_0 + 7$, y $\to Y_0 - 12$	$X_0 + Y_0 > 15$
$9 \to 11$	x $\to X_0 + 9$, y $\to Y_0 - 12$	$X_0 + Y_0 > 15$
$11 \to 14$	x $\to X_0 + 9$, y $\to Y_0 - 12$	$X_0 + Y_0 > 15 \wedge 2 * (X_0 + Y_0 - 3) \leq 21$
$14 \to 15$	x $\to X_0 + 9$, y $\to Y_0 - 12$	$X_0 + Y_0 > 15 \wedge 2 * (X_0 + Y_0 - 3) \leq 21$
$15 \to 16$	x $\to 4 * (X_0 + 9)$, y $\to Y_0 - 12$	$X_0 + Y_0 > 15 \wedge 2 * (X_0 + Y_0 - 3) \leq 21$
$16 \to 17$	x $\to 4*(X_0+9)$, y $\to 3*(Y_0-12)+4*(X_0+9)$	$X_0 + Y_0 > 15 \wedge 2 * (X_0 + Y_0 - 3) \leq 21$

Table 4: Symbolic Execution of Path4

Edge	Symbolic State (PV)	Path Condition (PC)
$1 \to 2$	x $\to X_0$, y $\to Y_0$	true
$2 \to 5$	x $\to X_0$, y $\to Y_0$	$X_0 + Y_0 \leq 15$
$5 \to 6$	x $\to X_0$, y $\to Y_0$	$X_0 + Y_0 \leq 15$
$6 \to 7$	x $\to X_0$ y $\to Y_0 + 10$	$X_0 + Y_0 \leq 15$
$7 \to 9$	x $\to X_0 - 2$, y $\to Y_0 + 10$	$X_0 + Y_0 \leq 15$
$9 \to 11$	x $\to X_0$, y $\to Y_0 + 10$	$X_0 + Y_0 \leq 15$
$11 \to 14$	x $\to X_0$, y $\to Y_0 + 10$	$X_0 + Y_0 \leq 15 \wedge 2 * (X_0 + Y_0 + 10) \leq 21$
$14 \to 15$	x $\to X_0$, y $\to Y_0 + 10$	$X_0 + Y_0 \leq 15 \wedge 2 * (X_0 + Y_0 + 10) \leq 21$
$15 \to 16$	x $\to 4 * X_0$, y $\to Y_0 + 10$	$X_0 + Y_0 \leq 15 \wedge 2 * (X_0 + Y_0 + 10) \leq 21$
$16 \to 17$	x $\to 4 * X_0$, y $\to 3 * (Y_0 + 10) + 4 * X_0$	$X_0 + Y_0 \leq 15 \wedge 2 * (X_0 + Y_0 + 10) \leq 21$

(c)

- Path1 is feasible, $X_0 = 10, Y_0 = 10$ is a feasible solution.
- Path2 is feasible, $X_0 = 5, Y_0 = 5$ is a feasible solution.
- Path3 is infeasible
- Path4 is feasible, $X_0 = 0, Y_0 = 0$ is a feasible solution.

2 Q2 Solution

(a) The "at-most-one" constraint can be encoded into an equivalent set of clauses in Conjunctive Normal Form (CNF) by considering all possible pairs of Boolean variables and ensuring that at most one variable in each pair is true. Here's the CNF representation for the at-most-one(a1, a2, a3, a4) constraint:

$$(\neg a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg a_3) \wedge (\neg a_1 \vee \neg a_4) \wedge (\neg a_2 \vee \neg a_3) \wedge (\neg a_2 \vee \neg a_4) \wedge (\neg a_3 \vee \neg a_4)$$

These clauses ensure that at most one of the Boolean variables (a1, a2, a3, a4) can be true at the same time. If any two or more of them are true, at least one of these clauses will be violated, satisfying the at-most-one constraint.
(b) The given FOL sentence **is valid**.
To establish the validity of the provided First Order Logic (FOL) sentence, we prove each direction of the biconditional independently. First, we prove the **forward direction**:

$$\forall x \exists y (P(x) \vee Q(y)) \implies (\forall x P(x)) \vee (\exists y Q(y))$$

Assume the left-hand side is true, indicating that for any x, there exists at least one y such that $P(x) \vee Q(y)$ holds. We aim to demonstrate that the right-hand side is also true.

1. **Case 1:** Suppose there exists a y_0 such that $Q(y_0)$ is true. Then for any x_0, we can choose y_0 (regardless of $P(x_0)$) such that $\exists y Q(y)$ is true and the right-hand side holds.
2. **Case 2:** Suppose there does not exist a y_0 such that $Q(y_0)$ is true. Based on the assumption that the left-hand side is true, we know that $\forall x P(x)$ must be true. Therefore, the right-hand side $(\forall x P(x)) \vee (\exists y Q(y))$ is also true.

In both cases, the right side holds when the left side is true, establishing the validity of the forward direction.
Now, we consider the **backward direction**:

$$(\forall x P(x)) \vee (\exists y Q(y)) \implies \forall x \exists y (P(x) \vee Q(y))$$

Assume the left-hand side is true, indicating that $(\forall x P(x)) \vee (\exists y Q(y))$ is true. We aim to demonstrate that the right-hand side is also true.

1. **Case 1:** If $\forall x P(x)$ is true, then for any x_0, we can choose any y_0 (regardless of $Q(y_0)$) such that $\exists y (P(x) \vee Q(y))$ holds.
2. **Case 2:** If $\exists y Q(y)$ is true, then for some y_0, $\exists y (P(x) \vee Q(y))$ holds for any x (regardless of $P(x)$).

In either case, the right-hand side holds, confirming the validity of the backward direction. Since both directions of the biconditional are valid, we conclude that the given FOL sentence **is valid**.
(c)

$$(\forall x \exists y(P(x,y) \vee Q(x,y))) \Rightarrow ((\forall x \exists y P(x,y)) \vee (\forall x \exists y Q(x,y)))$$

To show that this sentence is **not valid**, consider the following counterexample:
Let $M = \{p, q\}$, $P = \{(p, p)\}$, and $Q = \{(q, q)\}$.
For $x = p$, we can find $y = p$ such that $P(p, p)$ is true. Similarly, for $x = q$, we can find $y = q$ such that $Q(q, q)$ is true. The left side of the implication is satisfied:

$$\forall x \exists y(P(x,y) \vee Q(x,y)) \quad \text{is true}$$

However, the right side is not satisfied:

$$(\forall x \exists y P(x,y)) \vee (\forall x \exists y Q(x,y)) \quad \text{is false}$$

because when x = q, $P(q, y)$ is false for every y in {p ,q}, and when x = p, $Q(p, y)$ is also false for every y in {p, q}.
Therefore, there exists a model where the left side is true, but the right side is false, indicating that the given first-order logic sentence is not universally valid.
(d)

1. $M1 \models \Phi$. Suppose $x = 1, y = 3, z = 2$, we know that $x, y, z \in N$. Since $x < z < y$, we have P(x, y) is true, P(z, y) is true, P(x, z) is true and P(z, x) is false. Therefore, $M1 \models \Phi$.

2. $M2 \not\models \Phi$. Suppose $x, y, z \in N$, from $P(x, y) \wedge P(z, y)$, we know that $y = x + 1, y = z + 1$. Therefore, $x = z$. However, from $P(x, z)$, we know that $z = x + 1$, which contradicts with the previous conclusion that $x = z$. Therefore, we cannot find such $x \in N$, and $M2 \not\models \Phi$.

3. $M3 \models \Phi$. Suppose $x = \{1\}, y = \{1, 2, 3\}, z = \{1, 2\}$, we know that $x, y, z \in P(N)$. Since $x \subseteq y, z \subseteq y, x \subseteq z, z \not\subseteq x$, we have P(x, y) is true, P(z, y) is true, P(x, z) is true and P(z, x) is false. Therefore, $M3 \models \Phi$.

(e) The following solution was inspired by *Will Klieber et al. (2007). Efficient CNF Encoding for Selecting 1 from n Objects.*
Let $X = a_1, ..., a_n$, we then can divide the set of propositional variables X into m disjoint subsets denoted as G_1 through G_m. We then assign the label "c_i" to the commander node of group G_i. With this notation, the logic for the commander method can be represented in CNF form as outlined below.

1. At most one variable in a group can be true. For every group Gi, we encode the specified clauses as follows:

$$\bigwedge_{x_j \in G_i} \bigwedge_{x_k \in G_i, k<j} (\neg x_j \vee \neg x_k)$$

2. In the case where the commander variable of a group is false, no variables within the group can be true. We encode the following clauses for each group Gi:

$$\bigwedge_{x_j \in G_i} (c_i \vee \neg x_j)$$

3. At most one of the commander variables is true. This can be represented through a recursive application of the commander method.

Let's examine the number of clauses needed for each group. It's important to observe that Constraint 1, mentioned above, necessitates $\frac{k*(k-1)}{2}$ clauses, where k is the number of variables in the group. Constraint 2 requires k clauses. Consequently, the total number of clauses per group is calculated as $\frac{k*(k+1)}{2}$. Let's explore a grouping method in which, at each level of the hierarchy, the variables are divided into groups of size k. For instance, with k=2, we obtain the binary tree illustrated in Fig. 1. Examining the diagram reveals that, in the asymptotic limit of a large number of variables n, the number of groups is as follows:

$$\sum_{i=1}^{\infty}(n/k) = (n/k) + (n/k^2) + (n/k^3) + ... = (\frac{1/k}{1-1/k})n$$

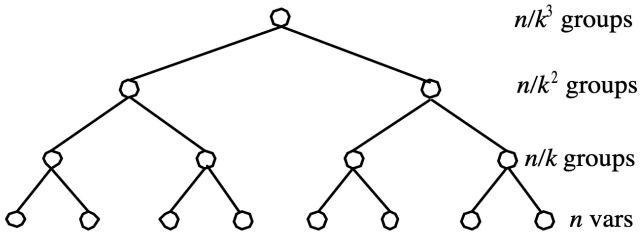

Figure 1: Binary tree of groups

The number of additional variables corresponds to the number of groups. To determine the total number of clauses, we multiply this by the clauses per group, resulting in a cumulative total of:

$$(\frac{(k+1)/2}{1-1/k})n$$

The optimal selection for minimizing the number of clauses is found to be k = 3. In this scenario, the total number of clauses is 3n, and the count of additional variables amounts to n/2. Therefore, this encoding uses at most O(n) clauses and variables.

3 Q3 Solution

To express quantifier-free constraints in First Order Logic (FOL) for a general n x n magic square, we can use the following constraints:
Let **Int(x)** be the predicate that check whether the object is an integer (true if x is an integer, false otherwise).

$$\bigwedge_{\substack{1\leq r_1,r_2,c_1,c_2\leq n \\ r_1\neq r_2 \text{ or } c_1\neq c_2}} a_{r_1c_1} \neq a_{r_2c_2}$$

$$\bigwedge_{1\leq r,c\leq n} Int(a_{rc})$$

$$\bigwedge_{1\leq r,c\leq n} 1 \leq a_{rc} \leq n^2$$

$$\bigwedge_{1\leq r\leq n} \sum_{1\leq c\leq n} a_{rc} = \frac{(1+n^2)\cdot n^2}{2}$$

$$\bigwedge_{1\leq c\leq n} \sum_{1\leq r\leq n} a_{rc} = \frac{(1+n^2)\cdot n^2}{2}$$

$$\bigwedge_{1\leq r\leq n} a_{rr} = \frac{(1+n^2)\cdot n^2}{2}$$

$$\bigwedge_{1\leq r\leq n} a_{r,n-r+1} = \frac{(1+n^2)\cdot n^2}{2}$$

4 Q4 Solution

(d) sym.py:132 is not covered because WHILE only has $'<='|'<'|'='|'>='|'>'$ relational operators.
sym.py:148 is not covered because WHILE only has $'not'|'and'|'or'$ logical operators.
sym.py:169 is not covered because WHILE only has $'+'|'-'|'*'|'/'$ arithmetic operators.
sym.py:322 is not covered because we did not execute the sym.py as the main program.
(e) The symbolic execution engine diverges for the following program

$$havoc\, x, y;\ while\, x > 0\, do\, \{while\, y > x\, do\, \{y := y/2 - 1\};\ x := x/2 - 1\}$$