Rylo Ashmore
HW 2

ECE 653: HOMEWORK 2

Problem 1. *Consider the following algorithm...*

```
havoc x,y;
if $x - y > 30$ then
    x=x+1;
    y=y-2
else
    y = y+5;
    x=x-3
end
;
if $3 * (x - y) < 40$ then
    x=x*3;
    y = y*2
else
    x = x*4;
    y = y*7
end
;
skip
```

...and the proposed modification of the conditional to `if x[i]>0 and x[i]%2==1`*. Answer the following:*

Problem 1.a. *How many execution paths does this program have? List all paths as a sequence of line numbers taken on the path as given in the original problem.*

(1,2,3,4,5,8,9,10,11,12,15)
(1,2,3,4,5,8,9,12,13,14.15)
(1,2,5,6,7,8,9,10,11,12,15)
(1,2,5,6,7,8,9,12,13,14,15)
are the execution paths. Not all are feasible.

Problem 1.b. *Symoblically execute each path and provide the resulting path condition. Show the steps of symbolic execution as a table.*

edge	symbolic state	path condition
$1 \to 2$	$x \mapsto X, y \mapsto Y$	true
$2 \to 3$	$x \mapsto X, y \mapsto Y$	$X - Y > 30$
$3 \to 4$	$x \mapsto X + 1, y \mapsto Y$	$X - Y > 30$
$4 \to 5$	$x \mapsto X + 1, y \mapsto Y - 2$	$X - Y > 30$
$5 \to 8$	$x \mapsto X + 1, y \mapsto Y - 2$	$X - Y > 30$
$8 \to 9$	$x \mapsto X + 1, y \mapsto Y - 2$	$X - Y > 30$
$9 \to 10$	$x \mapsto X + 1, y \mapsto Y - 2$	$X - Y > 30 \wedge 3 * (X + 1 - (Y - 2)) < 40$
$10 \to 11$	$x \mapsto 3X + 3, y \mapsto Y - 2$	$X - Y > 30 \wedge 3 * (X + 1 - (Y - 2)) < 40$
$11 \to 12$	$x \mapsto 3X + 3, y \mapsto 2Y - 4$	$X - Y > 30 \wedge 3 * (X + 1 - (Y - 2)) < 40$
$12 \to 15$	$x \mapsto 3X + 3, y \mapsto 2Y - 4$	$X - Y > 30 \wedge 3 * (X + 1 - (Y - 2)) < 40$

Second computation:

edge	symbolic state	path condition
$1 \to 2$	$x \mapsto X, y \mapsto Y$	true
$2 \to 3$	$x \mapsto X, y \mapsto Y$	$X - Y > 30$
$3 \to 4$	$x \mapsto X + 1, y \mapsto Y$	$X - Y > 30$
$4 \to 5$	$x \mapsto X + 1, y \mapsto Y - 2$	$X - Y > 30$
$5 \to 8$	$x \mapsto X + 1, y \mapsto Y - 2$	$X - Y > 30$
$8 \to 9$	$x \mapsto X + 1, y \mapsto Y - 2$	$X - Y > 30$
$9 \to 12$	$x \mapsto X + 1, y \mapsto Y - 2$	$X - Y > 30 \wedge 3 * (X + 1 - (Y - 2)) \geq 40$
$12 \to 13$	$x \mapsto 3X + 3, y \mapsto Y - 2$	$X - Y > 30 \wedge 3 * (X + 1 - (Y - 2)) \geq 40$
$13 \to 14$	$x \mapsto 4X + 4, y \mapsto Y - 2$	$X - Y > 30 \wedge 3 * (X + 1 - (Y - 2)) \geq 40$
$14 \to 15$	$x \mapsto 3X + 3, y \mapsto 7Y - 14$	$X - Y > 30 \wedge 3 * (X + 1 - (Y - 2)) \geq 40$

Different prefix now:

edge	symbolic state	path condition
$1 \to 2$	$x \mapsto X, y \mapsto Y$	true
$2 \to 5$	$x \mapsto X, y \mapsto Y$	$X - Y \leq 30$
$5 \to 6$	$x \mapsto X, y \mapsto Y$	$X - Y \leq 30$
$6 \to 7$	$x \mapsto X, y \mapsto Y + 5$	$X - Y \leq 30$
$7 \to 8$	$x \mapsto X - 3, y \mapsto Y + 5$	$X - Y \leq 30$
$8 \to 9$	$x \mapsto X - 3, y \mapsto Y + 5$	$X - Y \leq 30$
$9 \to 10$	$x \mapsto X - 3, y \mapsto Y + 5$	$X - Y \leq 30 \wedge 3 * (X - 3 - (Y + 5)) < 40$
$10 \to 11$	$x \mapsto 3X - 9, y \mapsto Y + 5$	$X - Y \leq 30 \wedge 3 * (X - 3 - (Y + 5)) < 40$
$11 \to 12$	$x \mapsto 3X - 9, y \mapsto 2Y + 10$	$X - Y \leq 30 \wedge 3 * (X - 3 - (Y + 5)) < 40$
$12 \to 15$	$x \mapsto 3X - 9, y \mapsto 2Y + 10$	$X - Y \leq 30 \wedge 3 * (X - 3 - (Y + 5)) < 40$

One final computation path:

edge	symbolic state	path condition
$1 \to 2$	$x \mapsto X, y \mapsto Y$	true
$2 \to 5$	$x \mapsto X, y \mapsto Y$	$X - Y \leq 30$
$5 \to 6$	$x \mapsto X, y \mapsto Y$	$X - Y \leq 30$
$6 \to 7$	$x \mapsto X, y \mapsto Y + 5$	$X - Y \leq 30$
$7 \to 8$	$x \mapsto X - 3, y \mapsto Y + 5$	$X - Y \leq 30$
$8 \to 9$	$x \mapsto X - 3, y \mapsto Y + 5$	$X - Y \leq 30$
$9 \to 12$	$x \mapsto X - 3, y \mapsto Y + 5$	$X - Y \leq 30 \wedge 3 * (X - 3 - (Y + 5)) \geq 40$
$12 \to 13$	$x \mapsto X - 3, y \mapsto Y + 5$	$X - Y \leq 30 \wedge 3 * (X - 3 - (Y + 5)) \geq 40$
$13 \to 14$	$x \mapsto 4X - 12, y \mapsto Y + 5$	$X - Y \leq 30 \wedge 3 * (X - 3 - (Y + 5)) \geq 40$
$14 \to 15$	$x \mapsto 4X - 12, y \mapsto 7Y + 35$	$X - Y \leq 30 \wedge 3 * (X - 3 - (Y + 5)) \geq 40$

Technically, every trace also has a final row of 15 $\to$ termination, but we ignore this as line 15 is simply a skip command.

Problem 1.c. *For each path above, indicate feasibility.*

The first path is infeasible.
$X = 0, Y = -31$ will execute the second path.
$X = 0, Y = 0$ will execute the third path.
$X = 0, Y = -11$ will execute the final path.
These results were found by inputting the final path conditions into a z3 solver.

Problem 2. *Consider some graphs.*

Problem 2.a. *The constraint $at - most - one(a_1, \dots, a_n)$ is satisfied if at most one of the Booleans a_i is true. Encode into an equivalent set of clauses.*

Clearly if there is at most one a_i true, then there is at most one a_i true pairwise. Similarly, if there is at most one a_i true pairwise, then there must be at most one a_i true overall (since a second would be ruled out by the pairing with the first). Hence we encode this as follows:

$$at - most - one(a_1, \dots, a_n) \equiv \bigwedge_{1 \leq i < j \leq n} (\neg a_i \vee \neg a_j)$$

Problem 2.b. *Let $G = (V, E)$ be a directed acyclic graph with vertices V, edges $E \subseteq V \times V$ and two vertices $v_{init}, v_{end} \in V$. Write a formula Path in CNF such that Path is sat iff there is a path from v_{init} to v_{end} in G.*

For each $v \in V$ and $e \in E$, create a boolean variable v and e. Intuitively, we want a satisfying assignment to correspond exactly to a path through V where true variables are those touched by the path.

For each $v \in V$, define $Out(v) = \{e \in E | e = (v, v')\}$ and $In(v) = \{e \in E | e = (v', v)\}$.

We then want v_{init} true so we have a start location. Next, we want $at-most-one(Out(v))$ true for all $v \in V$ to restrict each v to take one transition out (noting we already have this in CNF form). Next, we must indicate when an arbitrary intermediate state is reached. Either, it is not reached, or some incoming edge reaches it (excepting v_{init}). Thus we add on the clause

$$(\neg v \vee \bigvee_{e \in In(v)} e)$$

for each $v \in V - \{v_{init}\}$. Finally, we want to make sure the final state is reached. Simply adding v_{end} suffices for this. We also must say that if an edge is taken, then both vertices it connects are reached as well $((\neg e \vee v) \wedge (\neg e \vee v')$ for $e = (v, v'))$. Altogether, we need

$$v_{init} \wedge v_{end} \wedge \bigwedge_{v \in V - \{v_{init}\}} (\neg v \vee \bigvee_{e \in In(v)} e)) \wedge \bigwedge_{v \in V} at-most-one(Out(v)) \wedge \bigwedge_{e=(v,v')} (\neg e \vee v) \wedge (\neg e \vee v')$$

Suppose this formula is satisfied. Then for each $v \neq v_{init}$, we may find a preceding edge which is true. For this edge, it's preceding node must be true, and since v_{end} is true, we may recursively backtrack a trace until the only node without a predecessor, v_{init}.

Suppose G has a trace between the two points. Then the assignment v true iff v touched by the trace and e true iff e touched by the trace is a satisfying assignment.

Problem 2.c. *Extend $at-most-one(v_1, \ldots, v_n)$ to be linear in clauses and variables.*

Extra credit... is hard.

Problem 3. *SEND+MORE=MONEY. Solve the addition problem where each letter represents a unique digit.*

Given solution is 9567+1085=10652

Problem 3.a. *Write down quantifier free constraints in FOL w/ Arithmetic to solve the puzzle above.*

We print essentially the model generated by my Z3 python solver (formatted for LaTeX):

$$\begin{array}{ll}
(1) & \mathit{Distinct}(E, D, M, O, N, S, R, Y), \\
(2) & E >= 0, E <= 9, \\
(3) & D >= 0, D <= 9, \\
(4) & M >= 0, M <= 9, \\
(5) & O >= 0, O <= 9, \\
(6) & N >= 0, N <= 9, \\
(7) & S >= 0, S <= 9, \\
(8) & R >= 0, R <= 9, \\
(9) & Y >= 0, Y <= 9, \\
(10) & Or(D + E == Y, D + E == Y + 10), \\
(11) & If(D + E >= 10, \\
(12) & \quad Or(N + R + 1 == E, N + R + 1 == E + 10), \\
(13) & \quad Or(N + R == E, N + R == E + 10)), \\
(14) & If(N + R >= 10, \\
(15) & \quad Or(E + O + 1 == N, E + O + 1 == N + 10), \\
(16) & \quad Or(E + O == N, E + O == N + 10)), \\
(17) & If(E + O >= 10, \\
(18) & \quad Or(S + M + 1 == O, S + M + 1 == O + 10), \\
(19) & \quad Or(S + M == O, S + M == O + 10)), \\
(20) & If(S + M >= 10, \\
(21) & \quad Or(M == 1, M + 10 == 1), \\
(22) & \quad Or(M == 0, M + 10 == 0))
\end{array}$$

Here, line 1 is shorthand for the pair-wise conjunction that any two variables are not equal, lines 2-9 specify every variable is a digit, line 10 gives the rule for the least significant bit, and rules 11,14,17,20, give if-then-else rules for whether or not a carry occurs on the other four digits of the result. If there are ever more digits in the first two strings than the final string, we add separate constraints that any of those letters must be 0. For this case, these constraints to be added are empty.

Problem 3.b. *Implement a Z3 solver for problems of the form above.*

Done, including case-handling for any three-input sum problem, including handling input numbers bigger than output.

Problem 3.c. *Extend test suite*

Done, ICE+CREAM=CONE has no solution as C=0 by CREAM's first digit, then ICE+0REAM=0ONE makes R=0, a contradiction to Distinct(C,R).

Problem 4. *Do symoblic execution on WHILE language.*

Problem 4.a. *Do linear symbolic execution (no while or if statements)*

Done.

Problem 4.b. *Add in if statements.*

Done.

Problem 4.c. *Add in while statements.*

Done.

Problem 4.d. *Extend test suites to 100% branch coverage of parts a-c.*

Yup, 100% branch coverage (not on AExp as explained in README) on the class developed in parts a-c, SymExec.

Problem 4.e. *Give a test where the program runs slow.*

Given in test1.prg

Problem 5. *Exercises in FOL*

Problem 5.a. *Give a proof of validity for the following:*

$$(\forall x.\exists y.P(x) \vee Q(y)) \iff (\forall x.P(x) \vee \exists y.Q(y))$$

Proof. The inference rules from the slides do not include rules for $\iff$, so I will assume an informal proof suffices. We break this down into two cases, $lhs \implies rhs$ and $lhs \impliedby rhs$.

$(\forall x.\exists y.P(x) \vee Q(y)) \implies (\forall x.P(x) \vee \exists y.Q(y))$. Suppose that $\forall x.\exists y.(P(x) \vee Q(y))$. Suppose further, for sake of contradiction, that $\neg(\forall x.P(x) \vee \exists y.Q(y))$. Let c be an arbitrary element of the universe. By our first supposition, we know that $\exists y.(P(c) \vee Q(y)$. Let us name such a y, say $y = d$. Then $P(c) \vee Q(d)$. If $Q(d)$, then $\exists y.Q(y)$, and thus $\forall x.P(x) \vee \exists y Q(y)$, a contradiction with an earlier assumption. Hence $\neg Q(d)$, and thus $P(c)$ must be the other case. But recall c was arbitrary, and thus we conclude that $\forall x P(x)$. But now we conclude that $\forall x.P(x) \vee \exists y.Q(y)$, again a contradiction. In either case, we reach a contradiction, and thus, negating our assumption for contradiction, we conclude the desired result that $\forall x.P(x) \vee \exists y.Q(y)$.

$(\forall x.P(x) \vee \exists y.Q(y)) \implies (\forall x.\exists y.P(x) \vee Q(y))$. Suppose that $\forall x.P(x) \vee \exists y.Q(y)$. Let us take the first case. Suppose $\forall x.P(x)$. Then let c be an arbitrary element. Clearly $P(c)$. Further choose $y = c$ and observe $P(c) \vee Q(c)$. Then $\exists y.P(c) \vee Q(y)$. Finally, since c was arbitrary, we find that $\forall x.\exists y.P(x) \vee Q(y)$. Take the other case, that $\exists y.Q(y)$. Say $y = d$ is such a y that satisfies $Q(y)$. Let c be arbitrary again, and we choose $y = d$ to find that $Q(d)$ gives us that $P(c) \vee Q(d)$. Thus $\exists y.P(c) \vee Q(y)$, and since c is arbitrary, we conclude that $\forall x.\exists y.P(x) \vee Q(y)$. Thus in either case, we find our desired result, thus establishing it in general. □

Problem 5.b. *Disprove the following:*

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

Consider the universe of $\{a, b\}$ where $P = \{(a, a)\}$ and $Q = \{b, b\}$. We check that the hypothesis of the conditional is true and the conclusion false:

Hypothesis true: Let x be in the universe. If $x = a$, then choose $y = a$, and $P(a, a)$ is satisfied in the disjunction. If $x = b$, then choose $y = b$, and $Q(b, b)$ is satisfied in the disjunction. This satisfies all options for the universal quantifier, thus showing this claim true.

Conclusion false: Each disjunct is false:

$\forall x.\exists y.P(x, y)$ is false: Consider $x = b$. It is never the case that $P(b, y)$ is true, for $y = a$ or $y = b$. Thus the first disjunct is false.

$\forall x.\exists y.Q(x, y)$ is false: Consider $x = a$. It is never the case that $Q(a, y)$ is true, for $y = a$ or $y = b$. Thus the second disjunct is false.

Thus the disjunction of two falses is false, and true implies false is false, ergo we have made the overall statement false.

Problem 5.c. *Express in FOL that the unary predicate P has exactly two elements satisfying it.*

$$cardP2 \equiv \exists x.\exists y.(x \neq y \wedge P(x) \wedge P(y) \wedge \forall z.(P(z) \implies (z = x \vee z = y)))$$

In English, there are two things (x, y) which are distinct, both satisfy P, and anything in the world that satisfies P is one of those two.

Problem 5.d. *Express in FOL that location i is the pivot of array A such that any locations before i are smaller than any elements after i. Use the predicate $Array(\bullet)$, the function $len(\bullet)$, and the notation $\bullet[\bullet]$ for array reads.*

For writing this up, we will assume that $len(\bullet)$ is well defined even for non-array objects in the world to avoid type-ing issues (similarly, that $<$ is defined on objects other than integers, similarly, that $A[\bullet]$ is defined on non-integers. None of these cases should arise if we are smart about shortcutting truth evaluations, but we make this assumption to simplify our lives). If we are given array A, there is no need to test it is an array, as we know i is an index, and A is an array.

$$\forall j. \forall k. (int(j) \wedge int(k) \wedge 0 \leq j \wedge j < i \wedge i < k \wedge k < len(A)) \implies A[j] < A[k]$$

Note also that the classic definition of a pivot would say that $A[j] \leq A[k]$, while we are asked for strict inequality.

Problem 5.e. *Express that all elements of array A are distinct.*

Use the same notation and assumptions as above.

$$\forall i. \forall j. (int(i) \wedge int(j) \wedge 0 \leq i \wedge i < j \wedge j < len(A)) \implies A[i] \neq A[j]$$