CS536 Assignment 2

Due: Feb. 16th, 2025

Early Bird Due: Feb. 14th, 2025

(Ethics: Any behavior on any homework or exam that could be considered copying or cheating will result in an immediate zero on the assignment for all parties involved and will be reported to academichonesty@iit.edu. See the IIT Code of Academic Honesty, https://web.iit.edu/student-affairs/handbook/fine-print/code-academic-honesty)

1. Let u and v be some primitive type variables and α and β be some values that match the types of u and v (u and v might be the same variable, α and β might be the same value). When does $\sigma[u \mapsto \alpha][v \mapsto \beta] = \sigma[v \mapsto \beta][u \mapsto \alpha]$ and when does $\sigma[u \mapsto \alpha][v \mapsto \beta] \equiv \sigma[v \mapsto \beta][u \mapsto \alpha]$? Discuss the four different cases depending on whether $u \equiv v$ and whether $\alpha = \beta$.

		$\sigma[u \mapsto \alpha][v \mapsto \beta] = \sigma[v \mapsto \beta][u \mapsto \alpha]$?	$\sigma[u \mapsto \alpha][v \mapsto \beta] \equiv \sigma[v \mapsto \beta][u \mapsto \alpha]$?
$u \equiv v$	$\alpha = \beta$	YES	YES
$u \equiv v$	$\alpha \neq \beta$	NO	NO
$u \not\equiv v$	$\alpha = \beta$	YES	YES
$u \not\equiv v$	$\alpha \neq \beta$	YES	YES

Case 1: $u \equiv v$ and $\alpha = \beta$

Answer: YES for both equality and equivalence.

Reason: Since both variables are the same ($u \equiv v$) and both values are the same ($\alpha = \beta$), the order of substitution doesn't affect the final state. Substituting in any order results in the same state and the same functional outcome.

Case 2: $u \equiv v$ and $\alpha \neq \beta$

Answer: NO for both equality and equivalence.

Reason: Since u and v are the same variable but have different values, applying the substitution in different orders affects the result. The final state will depend on the substitution order, leading to different results for both equality and equivalence.

Case 3: $u \not\equiv v$ and $\alpha = \beta$

Answer: YES for both equality and equivalence.

Reason: Since u and v are different variables and the values are the same, substituting in either order results in the same state and functional outcome. The substitutions affect different variables, so the order does not matter.

Case 4: $u \not\equiv v$ and $\alpha \neq \beta$

Answer: YES for both equality and equivalence.

Reason: Since u and v are different variables and the values α and β are different, the order of substitution doesn't matter because the substitutions affect independent variables. Therefore, the state and the functional outcome will be the same regardless of the substitution order.

2. Consider state $\sigma = \{x = 5, y = 2\}$. Answer the following questions and show your work.
   a. Find state $\sigma[x \mapsto \sigma(y)][y \mapsto \sigma(x)]$.
   Step 1: $\sigma[x \mapsto \sigma(y)] = x = 2, y = 2$

   Step 2: Apply $\sigma[x \mapsto \sigma(y)][y \mapsto \sigma(x)] = x = 2, y = 2$

   So the result is:
   $\{x = 2, y = 2\}$

   b. Find state $\sigma[x \mapsto \sigma(y)][x \mapsto \sigma[x \mapsto \sigma(y)](x - y)]$.

   Step 1: $\sigma(x) = 5, \sigma(y) = 2$

   Step 2: Apply $\sigma[x \mapsto \sigma(y)]$ first: $\sigma[x \mapsto 2] = \{x = 2, y = 2\}$

   Step 3: Apply $\sigma[x \mapsto \sigma[y \mapsto 2](x - y)]$ next: $\sigma[x \mapsto 2 - 2] = \{x = 0, y = 2\}$

   So the result is:
   $\{x = 0, y = 2\}$

   c. Let $\tau = \sigma[x \mapsto 3]$, and $\gamma = \tau[y \mapsto \tau(x) * 4]$. Find γ.

   Step 1: $\tau = \sigma[x \mapsto 3] = \{x = 3, y = 2\}$
   Step 2: $\tau(x) = 3$

   Step 3: Apply $\gamma = \tau[y \mapsto \tau(x) * 4] = \tau[y \mapsto 3 * 4] = \tau[y \mapsto 12] = \{x = 3, y = 12\}$

   So the result is:
   $\{x = 3, y = 12\}$

3. Decide true or false for each of the following satisfaction. Justify your answers briefly.
   a. $\{x = 4, y = 2\}$ c $x < 3 \rightarrow y > 2$
   True

   Since $x < 3$ is false $4 !< 3$, the implication $x < 3 \rightarrow y > 2$ is always true based on the truth table of logical implication.

   b. $\{x = 3,\ b = (2,5,4,8)\}$ c $x > 1 \wedge \exists x.0 \leq x < 4 \wedge b[x] < 3$
   True

   Since $x > 1$ $(3 > 1)$ is true, and there exists $x = 0$ such that $0 \leq x < 4$ and $b[0] = 2 < 3$, the entire statement holds.

   c. σ c $\exists y. \forall 0 \leq x \leq 1. b[x] = y$, where $\sigma(b) = (2,2,3,3)$.
   True

Choosing $y = 2$, for $x = 0$ and $x = 1$, we have $b[x] = 2$, satisfying the universal quantification. Thus, the statement is true.

4. Translate the following Java programs into statements in our programming language.

   a. **for** (**int** $i = 0; i < b.length; i++$) $\{b[i] = i;\}$

   ```
   i := 0;
   while i < length(b) do
           b[i] := i;
           i := i + 1;
   od
   ```

   b. **while** $(x\ !\ =\ 1)$ $\{$**if** $(x\ \%\ 2 == 0)\ \{x = x/2;\}$ **else** $\{x++;\}\}$

   ```
   while x ≠ 1 do
           if x % 2 = 0 then
                   x := x / 2;
           else
                   x := x + 1;
           fi;
   od
   ```

   c. **int** $m = 8, p = 1, y = 1;$ **while** $(++m < 20)$ $\{\ p = p * (y++);\ \}$

   ```
   m := 8;
   p := 1;
   y := 1;
   while (m := m + 1) < 20 do
           p := p * (y + 1);
           y := y + 1;
   od
   ```

5. Evaluate each of the following configurations to completion. Do not use $\rightarrow^*$ or $\rightarrow^n$ in your solutions.

   a. $\langle$**if** $x < 2$ **then** $x := y + 1; w := x + 2$ **fi**, $\sigma\rangle$, where $\sigma = \{x = 3, y = 3, w = 4\}$

   $\sigma = \{x=3, y=3, w=4\}$

   Step 1:

   Evaluate the condition

   The condition is $x < 2$

   In the initial state, $x = 3$, which is **not less than 2**

   Step 2:

   Determine the effectSince the condition $x < 2$ is false, the `then` branch does not execute.

   Therefore, the assignments $x := y + 1$ and $w := x + 2$ are skipped.

   Step 3:

   Final state

   The state σ remains unchanged.

Final state: $\{x = 3, y = 3, w = 4\}$.

b. $\langle x := y + 1; y := x + 1,\ \sigma\rangle$. Here, variables x and y are both defined in state σ.
σ[x ↦ σ(y) + 1][y ↦ σ(y) + 2] (e.g., if σ(y)=2: {x=3, y=4})

Step 1: Evaluate the first assignment $x := y + 1$
Before the assignment, $y = 3$.
After $x := y + 1$, $x = 3 + 1 = 4$.

Step 2: Evaluate the second assignment $y := x + 1$
Now, $x = 4$ (from the previous assignment).
After $y := x + 1$, $y = 4 + 1 = 5$.

Step 3: Final state
The final state after both assignments is: $\{x = 4, y = 5\}$.

c. $\langle W, \sigma\rangle$, where $W \equiv$ **while** $x > y$ **do** S **od**, $S \equiv$ **if** $x > 0$ **then** $x := x + 1$ **else** $y := -2 * x$ **fi**, and σ c $y < x \leq 0$.
Terminates with x unchanged, y = -2 * x (e.g., σ: {x=0, y=0})

Step 1: Evaluate the while loop condition $x > y$
Initially, $x = 0$ and $y = 0$.
The condition $x > y$ is false because 0 > 0 is not true.

Step 2: Loop body execution
Since the while loop condition is false, the loop does not execute.
The state remains unchanged.

Step 3: Final state
The state after exiting the loop (without any iteration) is: $\{x = 0, y = 0\}$.
Terminated: No change in x, and y = -2 * x condition is not applicable here because the loop doesn't run.

6. Let $W \equiv$ **while** $x < 2$ **do** S **od**, where $S \equiv x := x + 1; p := p \wedge (x > 1)$.
   a. Calculate $M(S, \sigma_1)$. Here, σ_1 is a state with variables x and p defined.

Step 1: Execute the first assignment in S
$x := x + 1$

This updates x to $x_1 + 1$

Step 2: Execute the second assignment
$p := p \wedge (x > 1)$

This updates p based on whether $x_1 + 1 > 1$:
If $x_1 + 1 > 1$, then p remains p_1
If $x_1 \leq 1$, then p is updated to False

Therefore, the new state is:

$$M(S, \sigma_1) = x \mapsto x_1 + 1, p \mapsto p_1 \wedge (x_1 + 1 > 1)$$

b. Calculate $M(W, \sigma_2)$, where $\sigma_2(x) = 4$ and $\sigma_2(p) = T$.

Step 1: Check the while loop condition

The condition for the while loop is $x < 2$

In the initial state, $x = 4$, so $4 < 2$ is false, and the loop does not execute.

Step 2: Return the final state

Since the while loop condition is false at the start, the loop does not run, and the state remains unchanged:

$$M(W, \sigma_2) = \sigma_2 = x \mapsto 4, p \mapsto T$$

c. Calculate $M(W, \sigma_3)$, where σ_3 c $(x = 1) \wedge p$.

Step 1: Check the while loop condition

The condition for the while loop is $x < 2$

In the initial state, $x = 1$, so $1 < 2$ is true, and the loop will execute.

Step 2: First execution of S

Execute S:

$$S \equiv x := x + 1; p := p \wedge (x > 1)$$

$x := x + 1$ updates x from 1 to 2.

$p := p \wedge (x > 1)$ updates p to $T \wedge (2 > 1) = T$

Step 3: Check the while loop condition again

After the first iteration, $x = 2$

The condition $x < 2$ is now false, so the loop terminates.

Step 4: Return the final state

The final state is:

$M(W, \sigma_3) = x \mapsto 2, p \mapsto T$

7. Let $W \equiv \textbf{while}\ x > 0\ \textbf{do}\ S\ \textbf{od}$, where $S \equiv \textbf{if}\ x < y\ \textbf{then}\ x := y/x\ \textbf{else}\ x := x - 1; y := b[y]\ \textbf{fi}$. Answer the following questions.

   a. Calculate $M(S, \sigma_1)$ where $\sigma_1(x) = -2$ and $\sigma_1(y) = -1$.

```
while x > 0 do
        if x < y then
          x := y / x
          else
            x := x - 1
          y := b[y]
```

   {x=−2,y=−1}

   b. Calculate $M(W, \sigma_2)$ where $\sigma_2 = \{x = 2,\ y = 2,\ b = (0, 1, 2)\}$.

```
while x > 0 do
  if x < y then
    x := y / x
  else
    x := x - 1
  y := b[y]
```

   Incorrect, the program will enter an infinite loop.

   c. Calculate $M(W, \sigma_3)$ where $\sigma_3 = \{x = 8,\ y = 2,\ b = (4, 2, 0)\}$.

```
while x > 0 do
  if x < y then
    x := y / x
  else
    x := x - 1
  y := b[y]
```

   Correct, the program will throw an index out of bounds error.

8. In Lecture 5, without the consideration of $\perp$, we have the following conclusions:

$$\sigma \not\models \exists x \in S.\ p \Leftrightarrow \sigma\ \mathsf{c}\ \forall x \in S.\ \neg p$$
$$\sigma \not\models \forall x \in S.\ p \Leftrightarrow \sigma\ \mathsf{c}\ \exists x \in S.\ \neg p$$

   We used "$\sigma \not\models p \Leftrightarrow \sigma\ \mathsf{c}\ \neg p$" during the proof to get the above conclusions, so they are not totally correct anymore with the consideration of $\perp$. Let's create some conclusions without replacing $\sigma \not\models p$ with $\sigma\ \mathsf{c}\ \neg p$. Remember that the following definitions about satisfaction of quantified predicates are still correct (I rephrased them here):

   - $\sigma\ \mathsf{c}\ \exists x \in S.\ p$ means there exists $\alpha \in S$, it is the case that $\sigma[x \mapsto \alpha]\ \mathsf{c}\ p$.
   - $\sigma\ \mathsf{c}\ \forall x \in S.\ p$ means for every value $\alpha \in S$, it is the case that $\sigma[x \mapsto \alpha]\ \mathsf{c}\ p$.

   Fill in the blanks as required so that each of the following sentences is correct.

   - Fill $\underline{type\ 1}$ with one of the words: "*some*", "*every*" or "*this*".
   - Fill $\underline{type\ 2}$ with either "c"or "$\not\models$".

   a. $\sigma\ \mathsf{c}\ \exists x \in S.\ p$ means for $\underline{type\ 1}$ state σ and for $\underline{type\ 1}$ $\alpha \in S$, it is the case that $\sigma[x \mapsto \alpha]$ $\underline{type\ 2}$ p.

   some, $\models$

Explanation: For a given state, if there exists some value α such that p holds for that α, then the formula holds

b. σ c $\forall x \in S.\ p$ means for $\underline{type\ 1}$ state σ and for $\underline{type\ 1}$ $\alpha \in S$, it is the case that $\sigma[x \mapsto \alpha]$ $\underline{type\ 2}$ p.
every, ⊨

Explanation: For every value α, if p holds for all α, then the formula holds.

c. $\sigma \not\models \exists x \in S.\ p$ means for $\underline{type\ 1}$ state σ and for $\underline{type\ 1}$ $\alpha \in S$, it is the case that $\sigma[x \mapsto \alpha]$ $\underline{type\ 2}$ p.
every, ⊭

Explanation: For every value α, if p does not hold in a given state, the formula does not hold.

d. $\sigma \not\models \forall x \in S.\ p$ means for $\underline{type\ 1}$ state σ and for $\underline{type\ 1}$ $\alpha \in S$, it is the case that $\sigma[x \mapsto \alpha]$ $\underline{type\ 2}$ p.
some, ⊭

Explanation: For some values α, if p does not hold in a given state, the formula does not hold.

e. c $\exists x \in S.\ p$ means for $\underline{type\ 1}$ state σ, it is the case that σ $\underline{type\ 2}$ $\exists x \in S.\ p$.
every, ⊨

Explanation: For every state, if there exists a value α such that p holds in that state, then the formula holds.

f. c $\forall x \in S.\ p$ means for $\underline{type\ 1}$ state σ, it is the case that σ $\underline{type\ 2}$ $\forall x \in S.\ p$.
every, ⊨

Explanation: For every state, if p holds for all values α, then the formula holds.

g. $\not\models \exists x \in S.\ p$ means for $\underline{type\ 1}$ state σ, it is the case that σ $\underline{type\ 2}$ $\exists x \in S.\ p$.
some, ⊭

Explanation: For some state, if there exists a value α such that p holds in that state, then the formula holds.

h. $\not\models \forall x \in S.\ p$ means for $\underline{type\ 1}$ state σ, it is the case that σ $\underline{type\ 2}$ $\forall x \in S.\ p$.
some, ⊭

Explanation: For some state, if there exists a value α such that p does not hold in that state, the formula does not hold.