CS 536 - Science of Programming

Assignment - 2

1)

a) this, some, $\models$

b) this, every, $\models$

c) this, every, $\not\models$

d) this, some, $\not\models$

e) every, $\models$

f) every, $\models$

g) some, $\not\models$

h) some, $\not\models$

2)

a) $\{x=2, y=3\} \models x<2 \rightarrow y<3$

Here, $x=2$ & $y=3$

The condition says $x<2$, $y<3$. so, both are false

In implication, it is true if first part (or) left side is false regardless of the second part (or) right side of implication

$x<2$ is false

so, the statement is True

b) $\{b=(2,5,4,8)\} \models \exists m . 0 \le m < 4 \wedge b[m] < 2$

Here, it is given that m value is $0 \le m < 4$. so it is 0 to 3.

Given condition $b[m] < 2$

$\Rightarrow b[0] = 2$

$\Rightarrow b[1] = 5$

$\Rightarrow b[2] = 4$

$\Rightarrow b[3] = 8$

so it doesn't satisfy $b[m] < 2$

So, it is False

c) $\{x=2, b=(2,3)\} \models \exists y . \forall 0 \le x < 2 . b[x] = y$

For the above statement to be true, b[0] and b[1] must be equal to same value y. But, $b[0]=2$ & $b[1]=3$; So there is no single y that satisfies $b[x]=y$.

so, it is false

d) $\{x=1, b=(5,3,6)\} \models \forall x . \forall 0 \le k < 3 . x < b[k]$

Here, x is 1 and k is between 0 to 2

condition : $x < b[k]$

$1 < b[0] = 1 < 5$

$1 < b[1] = 1 < 3$

$1 < b[2] = 1 < 6$

Here the above condition is satisfied but in the statement it is given for all x, not just $x=1$. It is given as universal quantifier over x. so, if x is greater than 6 the statement does not satisfy the condition

so, it is false

3)

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

b) while $x \neq 1$ do if $x \% 2 = 0$ then $x := x/2$; else $x := x+1$; fi od

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

4)

a) Let $S \equiv$ if $x<2$ then $x := y+1$; $w := x+2$ fi

Now we have to evaluate S with state

$\sigma(x) = 3, \sigma(y) = 3, \sigma(w) = 4.$

$\langle S, \sigma \rangle = \langle$ if $x<2$ then $x := y+1$; $w := x+2$ fi, $\sigma \rangle$

$\rightarrow \langle$ skip, $\sigma \rangle$ // since $\sigma(x=3) < 2$ is False

$\rightarrow \langle E, \{x=3, y=3, w=4\} \rangle$

The final memory state remains $\{x=3, y=3, w=4\}$ as condition is false, Program terminated without changes.

b) Let $S \equiv$ while $x < 2$ do $x := y+1$; $w :=$ $x+2$ od

Now we need to evaluate S in state σ

$\sigma(x) = 1, \sigma(y) = 3, \sigma(w) = 4$

$\langle S, \sigma \rangle = \langle$ while $x < 2$ do $x := y+1$; $w := x+2$ od, $\{x = 1, y = 3, w = 4\} \rangle$

$\rightarrow \langle x := y+1; w := x+2$; while $x < 2$ do $x := y+1$; $w := x+2$ od, $\{x = 1, y = 3, w = 4\} \rangle$

// since $x = 1 < 2$ is T

$\rightarrow \langle w := x+2$; while $x < 2$ do $x := y+1$; $w := x+2$ od, $\{x \mapsto 4, y = 3, w = 4\} \rangle$

// $x := y+1 \Rightarrow x = 3+1 = 4$

$\rightarrow \langle$ while $x < 2$ do $x := y+1$; $w := x+2$ od, $\{x = 4, y = 3, w \mapsto 6\} \rangle$ // $w := x+2$
$\Rightarrow w = 4+2 = 6$

$\rightarrow \langle E, \{x = 4, y = 3, w = 6\} \rangle$ // since $x = 4 < 2$ is F,

The loop terminates

Final state

$\{x = 4, y = 3, w = 6\}$

The loop runs once because $x = 1$ is initially less than 2. After the first iteration,

x is updated to 4, which causes the loop to terminate.

c) Let $S \equiv x := y+1;\ y := x+1$.

Now we will need to evaluate S in state σ.

$\langle S, \sigma \rangle = \langle x := y+1;\ y := x+1, \sigma \rangle$

$\rightarrow \langle y := x+1, \sigma[x \mapsto \sigma(y)+1] \rangle$ // execute $x := y+1$, so x is updated to $\sigma(y)+1$

$\rightarrow \langle E, \sigma[x \mapsto \sigma(y)+1][y \mapsto (\sigma(y)+1)+1] \rangle$ // execute $y := x+1$, so y is updated to $\sigma(y)+2$

Final state :-

$\{x = \sigma(y)+1,\ y = \sigma(y)+2\}$

5) Let $S \equiv$ if $x > 0$ then $x := x+1$ else $y := -2 * x$ fi

Let $W \equiv$ while $x > y$ do S od

a) $\langle W, \sigma_1 \rangle$ where $\sigma_1 \models y < x \le 0$

$\langle W, \sigma_1 \rangle$

$\rightarrow \langle S; W, \sigma_1 \rangle$ // $\sigma_1(x) > \sigma_1(y)$

since $\sigma_1(x) > \sigma_1(y)$ holds, the loop body is executed.

$\rightarrow \langle y := -2 * x; W, \sigma_1 \rangle$ // $\sigma_1(x) \le 0$.

Since $\sigma_1(x) \leq 0$, the condition $x > 0$ is false and else branch is executed:
$y := -2 * x$..

$\rightarrow \langle W, \sigma_1 \{y \mapsto -2 * \sigma_1(x)\} \rangle$

The state is updated, setting y to $-2 * \sigma_1(x)$

$\rightarrow \langle E, \sigma_1 \{y \mapsto -2 * \sigma_1(x)\} \rangle$ // because, $\sigma_1(x) \leq 0$ & $\sigma_1\{y\} \geq 0$

Here, the loop terminates because after the update $x \leq y$ holds.

The final configuration:-

$$\langle E, \sigma_1 \{y \mapsto -2 * \sigma_1(x)\} \rangle$$

The final state is where y has been updated to $-2 * x$. & the loop terminates

b) $\langle W, \sigma_2 \rangle$ where $\sigma_2 \models x > 0 \wedge y \leq 0$

$$\langle W, \sigma_2 \rangle \rightarrow^* \langle W, \sigma_2 [x \mapsto \sigma_2(x) + 1] \rangle$$

Since, $\sigma_2(x) > 0$, the condition $x > y$ holds & the loop body S is executed. The then branch of S is triggered, updating x to $x+1$.

$\rightarrow^* \langle w, \sigma_2\{x \mapsto \sigma_2(x)+2\}\rangle$

The loop condition $x>y$ still holds, so the body s is executed again. x is incremented once more, now updated to $\sigma_2(x)+2$.

$\rightarrow^* \langle E, \bot_d\rangle$

The loop keeps incrementing x on each iteration. Since $x>y$ continues to hold, x binds to increasingly larger values in each iteration. The loop will either eventually terminate & lead more likely to divergence ($\bot_d$) where it keeps running indefinitely, as x keeps increasing with no bound.

The final configuration

$$\langle E, \bot_d\rangle$$

6) Let $w \equiv$ while $x<3$ do S od, where $S \equiv x := x+1; y := y*x$.

a) $M(S,\tau)$

$= \langle S,\tau\rangle = \langle x := x+1;\ y := y*x, \tau\rangle$

$\rightarrow \langle y := y*x, \tau\{x \mapsto \tau(x)+1\}\rangle$ // $x := x+1$

$\rightarrow \langle E, \tau\{x \mapsto \tau(x)+1, y \mapsto \tau(y)*(\tau(x)+1)\}\rangle$ // $y := y*x$

Thus, $M(S,\tau) = \tau\{x \mapsto \tau(x)+1, y \mapsto \tau(y)*(\tau(x)+1)\}$

b) $M(W, \sigma)$, where $\sigma(x) = 4$ and $\sigma(y) = 1$

$M(W, \sigma) = \langle W, \sigma \rangle = \langle$while $x < 3$ do S od$, \sigma \rangle$

$\rightarrow \langle E, \sigma \rangle$ // since $\sigma(x) = 4$ does'nt satisfy $x < 3$

Thus
$M(W, \sigma) = \{x = 4, y = 1\}$.

c) $M(W, \sigma)$, where $\sigma \models x = 1 \wedge y = 1$

$M(W, \sigma) = \langle W, \sigma \rangle = \langle$while $x < 3$ do S od$, \sigma \rangle$

$\rightarrow \langle S; W, \sigma[x \mapsto 2, y \mapsto 2] \rangle$ // $x := x+1$; $y := y * x$

$\rightarrow \langle S; W, \sigma[x \mapsto 3, y \mapsto 6] \rangle$ // $x := x+1$; $y := y * x$

$\rightarrow \langle E, \sigma[x \mapsto 3, y \mapsto 6] \rangle$ // Here the loop terminates since $x = 3$ & $x < 3$ is false.

Thus,

$M(W, \sigma) = \{x = 3, y = 6\}$

7) Let $w \equiv$ while $x > 0$ do s od, where $s \equiv$ if $x < y$ then $x := y / x$ else $x := x - 1$; $y := b[y]$ fi

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

$M(S, \sigma) = M(x := y/x, \sigma)$

$= M(E, \sigma[x \mapsto \sigma(y/x)])$

$= \{\sigma[x \mapsto 0]\}.$

Thus,

$M(S, \sigma) = \{\sigma[x \mapsto 0]\}.$

b) $M(w, \sigma)$ where $\sigma = \{x = 2, y = 2, b = (0,1,2)\}$.

$M(w, \sigma) = M(w, \{x = 2, y = 2, b = (0,1,2)\})$

$= M(w, \{x = 1, y = 2, b = (0,1,2)\})$

$= M(w, \{x = 2, y = 2, b = (0,1,2)\})$

$= \{\bot d\}.$

c) $M(w, \sigma)$ where $\sigma = \{x = 8, y = 2, b = (4,2,0)\}$.

$M(w, \sigma) = M(w, \{x = 8, y = 2, b = (4,2,0)\})$.

$= M(w, \{x = 7, y = 0, b = (4,2,0)\})$.

$= M(w, \{x = 6, y = 4, b = (4,2,0)\})$.

$= \{\bot e\}$. // array index out of bounds error occurs here.

d) No such state exists. The division y/x is possible only in the condition "if" is checked condition of statement S, & it is checked right after starting an iteration of W. As a result, whenever y/x is computed the condition $x>0$ must already hold so division by zero is impossible.

8) To have $M(S,\sigma) = \{\perp d\}$ the following one or more conditions must be true:

* $\alpha < 0$: This causes an error due to trying to compute the square root of a negative number.

* $\beta < 0$: This causes an error due to an invalid array index out-of-bounds access).

* $\beta \geq 4$: This also causes an error due to an out-of-bounds array index.

* $\beta = 1$: This causes a division by zero, as $b[1] = 0$.

So, the combination of above conditions will lead to a domain error ($\perp d$) when evaluating S.

9)

a) $\perp \models T$

True, because in logic T(true) is always satisfied regardless of the state or value, including undefined values like $\perp$.

so, $\perp \models T$ is True.

b) $\perp \not\models F$

True, because false (F) is not satisfied in the undefined value state ($\perp$).

Thus, $\perp \models F$ is false, which makes the statement $\perp \not\models F$ true.

c) if $\sigma(p) \neq \perp$, then $\sigma \models P$

False, because $\sigma(P) \neq \perp$ (P is well-defined in σ) doesn't necessarily mean that $\sigma \models P$ (that p is true). p could evaluate to a defined value but still be false.

So, the condition $\sigma(P) \neq \perp$ doesn't imply $\sigma \models P$.

d) if $\sigma(P) = \perp$, then $\not\models \neg P$

True, if $\sigma(P) = \perp$ it means that p is undefined & we can't conclude anything about the truth of P or $\neg P$.

Therefore, $\sigma \not\models \neg P$ because the truth value of P (& its negation $\neg p$) is indeterminate when p is undefined.

e) if $\models P$, then $\neg \exists \sigma . \sigma(P) = \bot$

True, if $\models P$ this means P is true in all states. For P to be true universally, it must be defined in all states, thus there cannot exist any state σ where $\sigma(P) = \bot$.

Therefore, the existence of a state where $\sigma(P) = \bot$ contradicts the fact that P is universally true.

10)

a) Let $\Sigma_0 \subset \Sigma$ and $\Sigma_0 \models P$, also let $\tau \models P$; then $\Sigma_0 \cup \{\tau\} \models P$

True,

If all states in Σ_0 satisfy P & τ also satisfies P then adding τ to Σ_0 will not change the fact that all states in the resulting set satisfy P. The union of two sets that both satisfy P will also satisfy P.

b) $\phi \models P$ and $\phi \models \neg P$

True, The empty set satisfies all predicates because there are no counter-examples. The statement "for all states in empty set, P is true" is true for any p including $\neg p$ because there are no state to check.

c) Let $\tau \in \Sigma$, then $\tau \models p$ or $\tau \models \neg p$

False, This statement assumes the law of excluded middle, which may not hold in all logical systems. In classical logic, this would be true, but in many-valued logics or in the presence of undefined values, a state might neither satisfy p nor $\neg p$.

d) Let $\Sigma_0 \cap \Sigma$, then $\Sigma_0 \models p$ or $\Sigma_0 \models \neg p$

False, This is not necessarily true for all subsets of Σ. There could be a subset Σ_0 where some states satisfy p and others satisfy $\neg p$. In such case, neither $\Sigma_0 \models p$ nor $\Sigma_0 \models \neg p$ would hold.

e) True, If σ_1 satisfies p_1 and σ_2 satisfies p_2, then the set $\{\sigma_1, \sigma_2\}$ satisfies the disjunction $p_1 \vee p_2$. This is because in every state in the set either p_1 or p_2 (or both) is true, which is the definition of disjunction.