ECS 240: HW 3

Donald Pinckney, 999845020

March 1, 2017

Problem 1

Part 1.1

Consider the new factorial program:

```
x := 5;
y := 1;
z := 2;
{while not (z = x + 1) do y := y * z; z := z + 1};
print y
```

By using a new counter variable, z, and incrementing z we can avoid the inaccuracy introduced with the subtraction. This converges to the fixed point of the else statement as $\{x := +, y := +, z := +\}$.

Part 1.2

I alter the abstract interpretation by choosing the abstract domain:

$$A = P(\{-, 0, +\}) \times P(\{-, 0, +\})$$

Note that $x := \bot$ is effectively $x := \emptyset$, and $x := \top$ is effectively $\{-, 0, +\}$. The ordering on A is given by lexicographical subset ordering. That is,

$$(U_1, V_1) \leq (U_2, V_2) \iff U_1 \subseteq U_2 \vee (U_1 = U_2 \wedge V_1 \subseteq V_2)$$

Instead of starting each point at $\bot$, we start each point at $(\emptyset, \emptyset)$, and the initial point at $(\{+\}, \{-, 0, +\})$. Then iterating the algorithm converges to a fixed point of $(x := \{0\}, y := \{+\})$ at the `true` branch.

Problem 2

A common technique for automated memory management is Automatic Reference Counting (ARC), recently popularized by the Clang compiler for Objective-C and Swift. The primary difficulty with ARC is the possibility of introducing retain cycles, where two or more objects create a cyclic memory dependency graph, which prevents deallocation of memory later. A possible application of AI could be to statically detect many of these retain cycles, where the abstract domain could model directed graph edges.

Problem 3

First I prove some small helper theorems:

Lemma 1. *Suppose S_1 and S_2 are subsets of a lattice A. Let $l = glb(S_1 \cup S_2)$. Then $l \leq glb(S_1) \wedge l \leq glb(S_2)$.*

Lemma Proof. For all $a \in S_1$, $l \leq a$, so l is a lower bound of S_1, and thus $l \leq glb(S_1)$. Likewise, for all $a \in S_2$, $l \leq a$, so l is a lower bound of S_2, so $l \leq glb(S_2)$. □

Lemma 2. *Suppose $S_1 \subseteq S_2 \subseteq A$, where A is a lattice. Then $glb(S_1) \geq glb(S_2)$.*

Lemma Proof. Note that $S_2 = S_1 \cup (S_2 \setminus S_1)$. Thus, by **Lemma 1** $glb(S_1) \geq glb(S_2)$. □

Now, for the actual problem:

Theorem 1. *Let C be the concrete domain, and A be the abstract domain, where A is a lattice. Let $\gamma : A \to P(C)$ be monotonic ($\forall a_1, a_2 \in A, a_1 \leq a_2 \implies \gamma(a_1) \leq \gamma(a_2)$). In addition, suppose γ has the property that for any $S \subseteq C$, there exist $b_1, b_2, \cdots, b_n \in A$ such that $S = \bigcup_{i=1}^{n} \gamma(b_i)$.*

Define $\alpha : P(C) \to A$ as $\alpha(S) = glb(\{a \mid \gamma(a) \supseteq S\})$. Then, α is monotonic and α and γ form a Galois Connection.

Proof. I begin by showing monotonicity of α. Let $S_1, S_2 \in P(C)$ be arbitrary, such that $S_1 \subseteq S_2$. Then, $S_1 \subseteq S_2 \implies \{a \mid \gamma(a) \supseteq S_1\} \supseteq \{a \mid \gamma(a) \supseteq S_2\}$. By **Lemma 2**,

$$\alpha(S_1) = glb(\{a \mid \gamma(a) \supseteq S_1\}) \leq glb(\{a \mid \gamma(a) \supseteq S_2\}) = \alpha(S_2).$$

Thus, α is monotonic.

Now I show that α and γ for a Galois Connection. First, I want to show that $\alpha(\gamma(a)) = a$. Expanding α gives:

$$\alpha(\gamma(a)) = glb(\{a' \mid \gamma(a') \supseteq \gamma(a)\}).$$

Now suppose that $a' \in \{a' \mid \gamma(a') \supseteq \gamma(a)\}$. Then, $\gamma(a') \supseteq \gamma(a)$, and by the monotonicity of γ, $a' \geq a$. Thus, $\forall a' \in \{a' \mid \gamma(a') \supseteq \gamma(a)\}, a' \geq a$, or in other words a is a lower bound of $\{a' \mid \gamma(a') \supseteq \gamma(a)\}$. In addition, $a \in \{a' \mid \gamma(a') \supseteq \gamma(a)\}$, so in fact

$$a = glb(\{a' \mid \gamma(a') \supseteq \gamma(a)\}) = \alpha(\gamma(a))$$

To show the reverse direction that $\gamma(\alpha(S) \supset S)$, start again by expanding α:

$$\alpha(S) = glb(\{a \mid \gamma(a) \supseteq S\})$$

Now, we make the substitution that there is the decomposition $S = \bigcup_{i=1}^{n} \gamma(b_i)$.

$$\begin{aligned}
\alpha(S) &= glb(\{a \mid \gamma(a) \supseteq S\}) \\
&= glb\left(\left\{a \mid \gamma(a) \supseteq \bigcup_{i=1}^{n} \gamma(b_i)\right\}\right) \\
&= glb\left(\left\{a \mid \bigwedge_{i=1}^{n} \gamma(a) \supseteq \gamma(b_i)\right\}\right) \\
&= glb\left(\bigcap_{i=1}^{n} \{a \mid \gamma(a) \supseteq \gamma(b_i)\}\right)
\end{aligned}$$

Now, since $\bigcap_{i=1}^{n} \{a \mid \gamma(a) \supseteq \gamma(b_i)\} \subseteq \{a \mid \gamma(a) \supseteq \gamma(b_j)\}$, for all j, by **Lemma 2** we have that for all j:

$$\alpha(S) = glb\left(\bigcap_{i=1}^{n} \{a \mid \gamma(a) \supseteq \gamma(b_i)\}\right) \geq glb(\{a \mid \gamma(a) \supseteq \gamma(b_j)\})$$

In addition, $glb(\{a \mid \gamma(a) \supseteq \gamma(b_j)\}) = b_j$, as otherwise a smaller glb would contradict the monotonicity of γ. Thus, $\alpha(S) \geq b_j$, for all j. By the monotonicity of γ, we have that $\gamma(\alpha(S)) \supseteq \gamma(b_j)$, for all j. Since $S = \bigcup_{i=1}^{n} \gamma(b_i)$, we finally obtain that $gamma(\alpha(S)) \supseteq \bigcup_{i=1}^{n} \gamma(b_i) = S$. Therefore, α and γ for a Galois Connection. $\square$

Problem 4

Part 4.1

Prove the following property using Hoare rules:

$$\{x \equiv 1 \pmod 2\}\ \texttt{while b do x := x + 2}\ \{x \equiv 1 \pmod 2\}$$

Proof. For concision, let $W = \texttt{while b do x := x + 2}$, and $P(x) = x \equiv 1 \pmod 2$. Also, note that clearly $P(x) \implies P(x+2)$. Then, the following proof tree proves the property $P(x)$ in the postcondition.

$$\dfrac{\vdash P(x) \implies P(x) \qquad \dfrac{\dfrac{\vdash P(x) \land b \implies P(x+2) \qquad \dfrac{}{\vdash \{P(x+2)\}\ \texttt{x := x + 2}\ \{P(x)\}} \qquad \vdash P(x) \implies P(x)}{\vdash \{P(x) \land b\}\ \texttt{x := x + 2}\ \{P(x)\}}}{\vdash \{P(x)\}\ W\ \{P(x) \land \neg b\}} \qquad \vdash P(x) \land \neg b \implies P(x)}{\vdash \{P(x)\}\ W\ \{P(x)\}}$$

$\square$

Part 4.2

Prove using Hoare rules the correctness of the following code:

$\{n \geq 0\}$

```
// Point 0
t := n;
r := 1;
// Point 1
while t != 0 do
    r := r * t;
    t := t - 1
end while
// Point 2
```

$\{r = n!\}$

Proof. First I start by proving a precondition to the while loop, at Point 1. In particular, I claim that $r(t!) = n!$ at Point 1. To see this, consider the proof tree:

$$\dfrac{\dfrac{\vdash n \geq 0 \implies \mathit{true} \equiv n = n \qquad \overline{\vdash \{n = n\}\ \texttt{t := n}\ \{t = n\}}}{\vdash \{n \geq 0\}\ \texttt{t := n}\ \{t = n\}} \qquad \dfrac{\vdash t = n \implies 1(t!) = n! \qquad \overline{\vdash \{1(t!) = n!\}\ \texttt{r := 1}\ \{r(t!) = n!\}}}{\vdash \{t = n\}\ \texttt{r := 1}\ \{r(t!) = n!\}}}{\vdash \{n \geq 0\}\ \texttt{t := n; r := 1}\ \{r(t!) = n!\}}$$

Thus, at Point 1 we have as a precondition (and as we will see, loop invariant) to the while loop the predicate $r(t!) = n!$. Now we must show that given the precondition $\{r(t!) = n!\}$, that the postcondition $\{r = n!\}$ holds at Point 2.

For concision, let $U =$ `r := r * t; t := t - 1`, and $W =$ `while` $\texttt{t} \neq \texttt{0}$ `do` U `end while`. Now, this proof tree completes the proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash r(t!) = n! \wedge t \neq 0 \implies rt(t!) = t(n!) \wedge t \neq 0 \qquad \overline{\vdash \{rt(t!) = t(n!) \wedge t \neq 0\}\ \texttt{r := r * t}\ \{r(t!) = t(n!) \wedge t \neq 0\}}}{\vdash \{r(t!) = n! \wedge t \neq 0\}\ \texttt{r := r * t}\ \{r(t!) = t(n!) \wedge t \neq 0\}} \qquad \dfrac{\vdash r(t!) = t(n!) \wedge t \neq 0 \implies r((t-1)!) = n! \qquad \overline{\vdash \{r((t-1)!) = n!\}\ \texttt{t := t - 1}\ \{r(t!) = n!\}}}{\vdash \{r(t!) = t(n!) \wedge t \neq 0\}\ \texttt{t := t - 1}\ \{r(t!) = n!\}}}{\vdash \{r(t!) = n! \wedge t \neq 0\}\ \texttt{r := r * t; t := t - 1}\ \{r(t!) = n!\}}}{\vdash \{r(t!) = n!\}\ \texttt{while t} \neq \texttt{0 do}\ U\ \{r(t!) = n! \wedge t = 0\}} \qquad \vdash r(t!) = n! \wedge t = 0 \implies r = n!}{\vdash \{r(t!) = n!\}\ W\ \{r = n!\}}$$

$\square$

Part 4.3

Prove that for any command c and assertion A, we can construct the derivation $\vdash \{A\}\ c\ \{true\}$.

Proof. To perform the proof, I use structural induction on the structure of c. Note that the inductive hypothesis is that for any A and any c, $\vdash \{A\}\ c\ \{true\}$.

Now I simply present each command, case by case:

1. `skip`

$$\dfrac{\dfrac{(\texttt{skip}\text{ rule})}{\vdash \{A\}\ \texttt{skip}\ \{A\}} \qquad \vdash A \implies true}{\vdash \{A\}\ \texttt{skip}\ \{true\}}$$

2. `x := e`

$$\dfrac{\vdash A \implies true \qquad \dfrac{(\text{Assignment rule, } [e/x]true \equiv true)}{\vdash \{true\}\ \texttt{x := e}\ \{true\}}}{\vdash \{A\}\ \texttt{x := e}\ \{true\}}$$

3. `c1; c2`

$$\dfrac{\dfrac{(\text{Inductive Hypothesis})}{\vdash \{A\}\ \texttt{c1}\ \{true\}} \qquad \dfrac{(\text{Inductive Hypothesis})}{\vdash \{true\}\ \texttt{c2}\ \{true\}}}{\vdash \{A\}\ \texttt{c1; c2}\ \{true\}}$$

4. `if b then c1 else c2`

$$\dfrac{\dfrac{(\text{Inductive Hypothesis})}{\vdash \{A \wedge b\}\ \texttt{c1}\ \{true\}} \qquad \dfrac{(\text{Inductive Hypothesis})}{\vdash \{A \wedge \neg b\}\ \texttt{c2}\ \{true\}}}{\vdash \{A\}\ \texttt{if b then c1 else c2}\ \{true\}}$$

5. `while b do c`

$$\dfrac{\vdash A \implies true \qquad \dfrac{\dfrac{(\text{Inductive Hypothesis})}{\vdash \{true \wedge b\}\ \texttt{c}\ \{true\}}}{\vdash \{true\}\ \texttt{while b do c}\ \{true \wedge \neg b\}} \qquad true \wedge \neg b \implies true}{\vdash \{A\}\ \texttt{while b do c}\ \{true\}}$$

This completes the structural induction, and thus for any A and c, $\vdash \{A\}\ c\ \{true\}$. □

Question 5

Lemma 3. *Let $\vdash'$ denote provability in Hoare Logic using an alternate rule of* `while`*. Then the alternate rule retains completeness of Hoare Logic if and only if*

$$\vdash' \{A\}\ \texttt{while b do c}\ \{A \land \neg b\}$$

is provable by applying the alternate rule and that $\vdash' \{A \land b\}\ c\ \{A\}$.

Proof. Suppose that for any P, Q, c, $\models \{P\}\ c\ \{Q\} \implies \vdash \{P\}\ c\ \{Q\}$. That is, Hoare Logic is complete under the original `while` rule.

We wish to show both directions of the biconditional statement:

1. ($\implies$). Suppose with an alternate rule of `while` that Hoare Logic is complete. Since $\vdash W :: \{A\}$ `while b do c` $\{A \land \neg b\}$ is provable with the original `while` rule, then by the soundness of Hoare Logic, $\models W$. By the assumption that with the alternate `while` rule Hoare Logic is complete, we also have that $\vdash' W$.

2. ($\impliedby$). Now, suppose that for some P, Q, c, $\models \{P\}\ c\ \{Q\}$. Then, we have that $\vdash \{P\}\ c\ \{Q\}$. In addition, we must have a corresponding derivation tree. To show that $\vdash' \{P\}\ c\ \{Q\}$ is provable, we perform structural induction on c. We use $\vdash' \{P\}\ c\ \{Q\}$ as the inductive hypothesis. There are two cases:

   (a) $c \not\equiv$ `while b do c1` $\lor$ $Q \not\equiv P \land \neg b$. Then, the root of the original derivation tree does **not** use the original while rule, and generically looks like:

   $$\frac{\dfrac{\cdots}{\vdash \alpha_1} \quad \dfrac{\cdots}{\vdash \alpha_2} \quad \cdots \quad \dfrac{\cdots}{\vdash \alpha_n}}{\vdash \{P\}\ c\ \{Q\}}$$

   For the derivation tree of $\vdash' \{P\}\ c\ \{Q\}$ we choose:

   $$\frac{\dfrac{\cdots}{\vdash' \alpha_1} \quad \dfrac{\cdots}{\vdash' \alpha_2} \quad \cdots \quad \dfrac{\cdots}{\vdash' \alpha_n}}{\vdash' \{P\}\ c\ \{Q\}}$$

   where $\vdash' \alpha_i$ are provable by the inductive hypothesis.

   (b) $c \equiv$ `while b do c1` $\land$ $Q \equiv P \land \neg b$. Then, the root of the original derivation tree looks like:

   $$\frac{\dfrac{\cdots}{\vdash \{P \land b\}\ \texttt{c1}\ \{P\}}}{\vdash \{P\}\ \texttt{while b do c1}\ \{P \land \neg b\}}$$

   Now, by assumption we have that $\vdash' \{P\}$ `while b do c1`$\{P \land \neg b\}$ is provable using the alternate rule and $\vdash' \{A \land b\}\ c1\ \{A\}$. In addition, $\vdash' \{A \land b\}\ c1\ \{A\}$ is provable by the inductive hypothesis. Thus, $\vdash' \{P\}\ c\ \{Q\}$ is provable.

   This completes the structural induction, and thus we have shown that if $\vdash' \{A\}$ `while b do c` $\{A \land \neg b\}$ is provable using the alternate rule and that $\vdash' \{A \land b\}\ c\ \{A\}$, then Hoare Logic with the alternate rule is also complete.

□

Now we need to apply **Lemma 3** to each possible alternate rule.

Part 5.1

The alternate rule is:

$$\frac{\vdash A \land b \implies C \qquad \vdash \{C\}\ c\ \{A\} \qquad \vdash A \land \neg b \implies B}{\vdash \{A\}\ \texttt{while b do c}\ \{B\}}$$

We can use this rule to prove $\vdash \{A\}$ `while b do c` $\{A \land \neg b\}$ by choosing $B \equiv A \land \neg b$ and $C \equiv A \land b$:

$$\frac{\vdash A \land b \implies A \land b \qquad \vdash \{A \land b\}\ c\ \{A\} \qquad \vdash A \land \neg b \implies A \land \neg b}{\vdash \{A\}\ \texttt{while b do c}\ \{A \land \neg b\}}$$

By **Lemma 3** this alternate rule retains completeness.

Part 5.2

The alternate rule is:

$$\frac{\vdash \{A\}\ c\ \{b \implies A \wedge \neg b \implies B\}}{\vdash \{b \implies A \wedge \neg b \implies B\}\ \texttt{while b do c}\ \{B\}}$$

Now, we wish to prove $\vdash \{A'\}$ `while b do c` $\{A' \wedge \neg b\}$. Though this is more tricky, if we choose $A \equiv A' \wedge b$, $B \equiv A' \wedge \neg b$, we can complete the proof:

$$\frac{\vdash A' \overset{\textcircled{1}}{\implies} [(b \implies A' \wedge b) \wedge (\neg b \implies A' \wedge \neg b)] \qquad \dfrac{\dfrac{\vdash A' \wedge b \implies A' \wedge b \qquad \vdash \{A' \wedge b\}\ c\ \{A'\} \qquad \vdash A' \overset{\textcircled{1}}{\implies} (b \implies A' \wedge b) \wedge (\neg b \implies A' \wedge \neg b)}{\vdash \{A' \wedge b\}\ c\ \{(b \implies A' \wedge b) \wedge (\neg b \implies A' \wedge \neg b)\}}}{\vdash \{(b \implies A' \wedge b) \wedge (\neg b \implies A' \wedge \neg b)\ \texttt{while b do c}\ \{A' \wedge \neg b\}\}} \qquad \vdash A' \wedge \neg b \implies A' \wedge \neg b}{\vdash \{A'\}\ \texttt{while b do c}\ \{A' \wedge \neg b\}}$$

Finally, I quickly prove $\textcircled{1}$. Consider the case when $A' = \mathit{true}$. Then,

$$\begin{aligned}
&\mathit{true} \implies (b \implies \mathit{true} \wedge b) \wedge (\neg b \implies \mathit{true} \wedge \neg b) \\
\equiv\ &\mathit{true} \implies (b \implies b) \wedge (\neg b \implies \neg b) \\
\equiv\ &\mathit{true}
\end{aligned}$$

Thus, $\textcircled{1}$ is proved, so $\vdash \{A'\}$ `while b do c` $\{A' \wedge \neg b\}$ is provable, and by **Lemma 3** this alternate rule retains completeness.

Part 5.3

The alternate rule is:

$$\frac{\vdash \{A \wedge b\}\; c\; \{A\}}{\vdash \{A\}\; \texttt{while b do c}\; \{A\}}$$

This rule does **NOT** retain completeness. Although it has the fundamental concept of a loop invariant correct, it looses the information about the termination of the loop via the negation of the loop condition. To see this concretely, I prove that it is impossible to derive a proof of $\{A \wedge b\}$ `while b do c` $\{A \wedge \neg b\}$. Suppose it is provable, so we will try to construct a derivation tree. Since it is not in the form of the rule, the **only** option is to apply the Rule of Consequence to get it into the form of the rule:

$$\frac{\vdash A \wedge b \implies A' \qquad \vdash \{A'\}\; \texttt{while b do c}\{A'\} \qquad \vdash A' \implies A \wedge \neg b}{\vdash \{A \wedge b\}\; \texttt{while b do c}\; \{A \wedge \neg b\}}$$

For some predicate A'. However, regardless of what A' is, this leads to a contradiction:

$$A \wedge b \implies A' \implies A \wedge \neg b$$

And thus $b \implies \neg b$, which is clearly a contradiction. Thus, we can not derive a proof of $\{A \wedge b\}$ `while b do c` $\{A \wedge \neg b\}$, even though this would be trivial with the original while rule. Therefore, this alternate rule does **NOT** retain completeness.

Part 5.4

The alternate rule is:

$$\frac{\vdash \{A\}\; c\; \{A\}}{\vdash \{A\}\; \texttt{while b do c}\; \{A \wedge \neg b\}}$$

This rule does **NOT** retain completeness. It needlessly strengthens the loop invariant to be invariant not only during the execution of the loop, but over states in which the loop wouldn't execute. In some cases, depending on the c, this may be impossible to satisfy.

For an example, consider $\vdash \{i \geq 0\}$ `while i` $\geq$ `0 do i := i-1` $\{i = -1\}$. While this is easy to prove using the original rule, it is impossible to prove with this alternate rule. Ultimately when trying to prove this, one will encounter:

$$\{i \geq -1\}\; \texttt{i := i-1}\; \{i \geq -1\}$$

which is impossible to prove because it simply isn't true.