Exercises 4

Exercise 1 (Weakest Prediction)**.** Recall the definition of weakest precondition:

$$\mathrm{wp}(r, Q) = \{s \mid \forall s'.\ (s, s') \in r \to s' \in Q\}$$

1. Prove or disprove the following properties:
   - $\mathrm{wp}(r_1 \cup r_2, Q) = \mathrm{wp}(r_1, Q) \cup \mathrm{wp}(r_2, Q)$
   - $\mathrm{wp}(r_1 \cup r_2, Q) = \mathrm{wp}(r_1, Q) \cap \mathrm{wp}(r_2, Q)$
   - $\mathrm{wp}(r, Q_1 \cap Q_2) = \mathrm{wp}(r, Q_1) \cap \mathrm{wp}(r, Q_2)$
   - $\mathrm{wp}(r, Q_1 \cup Q_2) = \mathrm{wp}(r, Q_1) \cup \mathrm{wp}(r, Q_2)$

   For those that are wrong, do any of them hold if r is restricted to being functional, i.e. if r satisfies

   $$\forall x, y_1, y_2.\ ((x, y_1) \in r \land (x, y_2) \in r) \to (y_1 = y_2)$$

2. Let $r \subseteq S \times S$ and $Q \subseteq S$. Give an expression defining weakest precondition $\mathsf{wp}(r, Q)$ using operations of inverse of a relation, $(\_^{-1})$, set difference $(\setminus)$, and image of a relation under a set, $(\_[\_])$. Prove that your expression is correct by expanding the definitions of wp as well as of relational and set operations.

Solution:

- *The first statement is false:*

  Let $r_1 = \{(a, a), (b, a)\}$, $r_2 = \{(a, b), (b, b)\}$ *and* $Q = \{b\}$.

  Then $wp(r_1, Q) = \emptyset$, $wp(r_2, Q) = \{a, b\}$ *and* $wp(r_1 \cup r_2, Q) = \emptyset$.

  Note that r_1, r_2 *and* $r_1 \cup r_2$ *are all functional relations.*

- *The second statement is true:*

$$\begin{aligned} wp(r_1 \cup r_2, Q) &= \{s \mid \forall s'.\ (s,s') \in r_1 \cup r_2 \to s' \in Q\} \\ &= \{s \mid \forall s'.\ (s,s') \in r_1 \vee (s,s') \in r_2 \to s' \in Q\} \\ &= \{s \mid \forall s'.\ (s,s') \in r_1 \to s' \in Q \wedge (s,s') \in r_2 \to s' \in Q\} \\ &= \{s \mid (\forall s'.\ (s,s') \in r_1 \to s' \in Q) \wedge (\forall s'.\ (s,s') \in r_2 \to s' \in Q)\} \\ &= \{s \mid \forall s'.\ (s,s') \in r_1 \to s' \in Q\} \cap \{s \mid \forall s'.\ (s,s') \in r_2 \to s' \in Q\} \\ &= wp(r1, Q) \cap wp(r2, Q) \end{aligned}$$

- *The third statement is also true:*

$$\begin{aligned} wp(r, Q_1 \cap Q_2) &= \{s \mid \forall s'.\ (s,s') \in r \to s' \in Q_1 \cap Q_2\} \\ &= \{s \mid \forall s'.\ (s,s') \in r \to (s' \in Q_1 \wedge s' \in Q_2)\} \\ &= \{s \mid \forall s'.\ ((s,s') \in r \to s' \in Q_1) \wedge ((s,s') \in r \to s' \in Q_2)\} \\ &= \{s \mid (\forall s'.\ (s,s') \in r \to s' \in Q_1) \wedge (\forall s'.\ (s,s') \in r \to s' \in Q_2)\} \\ &= \{s \mid \forall s'.\ (s,s') \in r \to s' \in Q_1\} \wedge \{s \mid \forall s'.\ (s,s') \in r \to s' \in Q_2\} \\ &= wp(r, Q_1) \cap wp(r, Q_2) \end{aligned}$$

- *The fourth statement is true only when r is functional:*

  Let $r = \{(a,b),(a,c)\}, Q_1 = \{b\}, Q_2 = \{c\}$.

  We have $wp(r, Q_1) = wp(r, Q_2) = \{b, c\}$ *but* $wp(r, Q_1 \cup Q_2) = \{a, b, c\}$.

  When r is a partial function the weakest precondition becomes the union between the preimage and the set of all the elements that are not in the domain. This is due to the fact that when r is a partial function the image of an element is either a singleton or the empty set. Let s be an arbitrary element, if s is not in the domain then

$$\forall s'.\ (s,s') \in r \to s' \in Q$$

  is true as the condition of the implication is always false. If s is in the domain then the statement reduces to whether its image is in Q, which is equivalent to $\exists s'.\ (s,s') \in r \wedge s' \in Q$. *The set of elements verifying this property is the preimage of Q under r also noted* $r^{-1}[Q]$. *Therefore we have* $wp(r, Q) = r^{-1}[Q] \cup \{s \mid \forall s'.\ (s,s') \notin r\}$

We first show that the preimage of an union is the union of the preimages.

$$\begin{aligned} r^{-1}[Q_1 \cup Q_2] &= \{s \mid \exists s'.\ (s,s') \in r \wedge s' \in Q_1 \cup Q_2\} \\ &= \{s \mid \exists s'.\ (s,s') \in r \wedge (s' \in Q_1 \vee s' \in Q_2)\} \\ &= \{s \mid \exists s'.\ ((s,s') \in r \wedge s' \in Q_1) \vee ((s,s') \in r \wedge s' \in Q_2)\} \\ &= \{s \mid (\exists s'.\ (s,s') \in r \wedge s' \in Q_1) \vee (\exists s'.\ (s,s') \in r \wedge s' \in Q_2)\} \\ &= \{s \mid \exists s'.\ (s,s') \in r \wedge s' \in Q_1\} \cup \{s \mid \exists s'.\ (s,s') \in r \wedge s' \in Q_2\} \\ &= r^{-1}[Q_1] \cup r^{-1}[Q_2] \end{aligned}$$

We thus have:

$$\begin{aligned} wp(r, Q_1 \cup Q_2) &= r^{-1}[Q_1 \cup Q_2] \cup \{s \mid \forall s'.\ (s,s') \notin r\} \\ &= r^{-1}[Q_1] \cup r^{-1}[Q_2] \cup \{s \mid \forall s'.\ (s,s') \notin r\} \\ &= wp(r, Q_1) \cup wp(r, Q_2) \end{aligned}$$

The last step uses the fact that the set of elements which are not in the domain do not depend on Q_1 or Q_2

Alternative proof when r is functional is as follows. First, observe that, in general

$$wp(r, Q) = \{s \mid r[\{s\}] \subseteq Q\}$$

When r is functional then each $r[\{s\}]$ is either empty set or a singleton set. Next, when C is an empty or singleton sets, then

$$C \subseteq A \cup B \quad \Longleftrightarrow \quad (C \subseteq A \ \vee \ C \subseteq B)$$

Putting all these together, we have that, for functional r,

$$\begin{aligned} wp(r, Q_1 \cup Q_2) &= \{s \mid r[\{s\}] \subseteq Q_1 \cup Q_2\} \\ &= \{s \mid r[\{s\}] \subseteq Q_1 \vee r[\{s\}] \cup Q_2\} \\ &= \{s \mid r[\{s\}] \subseteq Q_1\} \cup \{s \mid r[\{s\}] \subseteq Q_2\} \\ &= wp(r, Q_1) \cup wp(r, Q_2) \end{aligned}$$

$\Diamond$

Exercise 2 (Programing With Integers)**.** Consider the following program:

```
1  case class Container(var x: INT, var y: INT):
2    def fun: Unit = {
3      require(x > 0 && y > 0)
4      if x > y then
5        x = x+y
```

```
6           y = x−y
7           x = x−y
8         else
9           y = y−x
10      }.ensuring(2∗old(this).x + old(this).y > 2∗x + y &&
11                   x >= 0 && y >= 0)
12    end Container
```

- Compute $R(\text{fun})$ formally, by expressing all intermediate formulas corresponding to subprograms.
- Write the formula expressing the correctness of the ensuring clause. Is the formula valid when INT denotes mathematical integers with their usual operations (**type** INT = BigInt in Scala)?
- Is the the verification condition formula valid for machine integers,

  $$Z_{2^{32}} = \{-2^{31}, \ldots, -1, 0, 1 \ldots, 2^{31} - 1\}$$

  and where operations are intepreted as the usual machine arithmetic (**type** INT = Int in Scala)?

Solution:

- *Using quite mechanically the following:*

Command	*Formula*
$f; s$	$\exists x_i, y_i.\ R(f)\,[x', y' := x_i, y_i] \wedge R(s)\,[x, y := x_i, y_i]$
if c ***then*** t ***else*** e	$(R(c) \wedge R(t)) \vee (\neg R(c) \wedge R(e))$

  We get:

Command	*Formula*
$x = x+y$	$F_1 := x' = x + y \wedge y' = y$
$y = x-y$	$F_2 := x' = x \wedge y' = x - y$
$x = x-y$	$F_3 := x' = x - y \wedge y' = y$
$y = y-x$	$F_4 := x' = x \wedge y' = y - x$
if body	$F_{\text{if}} := \exists x_{i1}, y_{i1}, x_{i2}, y_{i2}.$ $F_1\,[x', y' := x_{i1}, y_{i1}] \wedge$ $F_2\,[x, y, x', y' := x_{i1}, y_{i1}, x_{i2}, y_{i2}] \wedge$ $F_3\,[x, y := x_{i2}, y_{i2}]$
else body	$F_{\text{else}} := F_4$
program	$F := ((x > y) \wedge F_{\text{if}}) \vee (\neg(x > y) \wedge F_{\text{else}})$

- *We first express the pre- and post-condition:*

  $$P = x > 0 \wedge y > 0$$
  $$Q = 2x + y > 2x' + y' \wedge x' > 0 \wedge y' > 0$$

We can then express the correctness formula:

$$\forall x, y, x', y'.\ (P \wedge F) \to Q$$

One can conclude that this function is correct by realizing that the function amounts to:

- *If $x > y$, swap x and y;*
- *Else, decrease y by x.*

and then doing case analysis.

- *We note $\oplus$ for wrapping addition (e.g. $(2^{31} - 1) \oplus 1 = -2^{31}$).*

  The body of the program is still correct, however the postcondition doesn't hold: consider $x = 1,\ y = 2^{31} - 1$. Then $x' = 1,\ y' = 2^{31} - 2$. The first postcondition is

  $$2^{31} \oplus 1 = 2 \oplus (2^{31} - 1) = 2x \oplus y \overset{?}{>} 2x' \oplus y' = 2 \oplus (2^{31} - 2) = 2^{31}$$

  However, due to wrapping, $2^{31} \oplus 1 = -2^{31} < 0$.

  If you want to test this with stainless, you have to use $--strict-arithmetic=false$.

$\Diamond$

Exercise 3 (Hoare Logic Proof)**.** Give a complete Hoare logic proof for the following program:

```
{n >= 0 && d > 0}
  q = 0
  r = n
  while ( r >= d ) {
    q = q + 1
    r = r - d
  }
{n == q * d + r && 0 <= r < d}
```

The proof should include step-by-step annotation for each line of the program, as in the example proof in the lecture. **Solution:**

```
// {n >= 0 && d > 0}
q = 0
// {n >= 0 && d > 0 && q == 0}
r = n
// {n >= 0 && d > 0 && q == 0 && r == n}
while // {d > 0 && n == q * d + r && 0 <= r}
(r >= d) {
// {d > 0 && n == q * d + r && d <= r}
q = q + 1
// {d > 0 && n == (q-1) * d + r && d <= r}
r = r - d
// {d > 0 && n == (q-1) * d + r + d && 0 <= r}
// {d > 0 && n == q * d + r && 0 <= r}
}
// {d > 0 && n == q * d + r && 0 <= r && r < d}
// {n == q * d + r && 0 <= r < d}
```

$\Diamond$

Exercise 4 (Iterating a Relation)**.** Let $M = (S, I, r, A)$ be a transition system and $\bar{r} = \{(s, s') \mid (s, a, s') \in r\}$, as usual. Let $\Delta = \{(x, x) \mid x \in S\}$.

Let $\bar{r}^k$ denote the usual composition of relation $\bar{r}$ with itself k times.

Define the sequence of relations r_n, for all non-negative integers n, as follows:

- $r_0 = \Delta \cup \bar{r}$
- $r_{n+1} = r_n \circ r_n$

A) (2pt) Prove that $r_n \subseteq r_{n+1}$ for every n. **Solution:** *By induction:*

Base case:

$$\begin{aligned} r_0 \circ r_0 &= (\Delta \cup \bar{r}) \circ (\Delta \cup \bar{r}) \\ &= (\Delta \circ \Delta) \cup (\Delta \circ \bar{r}) \cup (\bar{r} \circ \Delta) \cup (\bar{r} \circ \bar{r}) \\ &= \Delta \cup \bar{r} \cup \bar{r}^2 \end{aligned}$$

In the previous steps we used the fact that composition distributes over union (cf Exercise Set 2 Part II), and that Δ is a neutral element wrt composition (since $\Delta = \bar{r}^0$).

$$r_0 = \Delta \circ \bar{r} \subseteq \Delta \cup \bar{r} \cup \bar{r}^2 = r_0 \circ r_0 = r_1$$

Induction:

$$r_{n+1} = r_n \circ r_n \subseteq r_{n+1} \circ r_{n+1} = r_{n+2}$$

By IH and the fact that if $r_1 \subseteq r_1'$ and $r_2 \subseteq r_2'$ then $r_1 \circ r_2 \subseteq r_1' \circ r_2'$

$\Diamond$

B) (3pt) Prove that for every n and every k where $0 \leq k \leq 2^n$ we have $\bar{r}^k \subseteq r_n$. **Solution:** *By induction:*

Base case:

$$\bar{r}^0 = \Delta \subseteq \Delta \cup \bar{r} = r_0 \quad \bar{r}^1 = \bar{r} \subseteq \Delta \cup \bar{r} = r_0$$

Induction:

If $0 \leq k \leq 2^n$: $\bar{r}^k \subseteq r_n \subseteq r_{n+1}$ by IH and A)

If $2^n \leq k \leq 2^{n+1}$: $\bar{r}^k = r^{2^n} \circ r^{k-2^n} \subseteq r_n \circ r_n = r_{n+1}$ by IH and since $0 \leq k - 2^n \leq 2^n$ $\Diamond$

C) (2pt) Suppose that S is finite. Find a bound B as a function of $|S|$ such that

$$\mathsf{Reach}(M) \subseteq r_B[I]$$

Aim to find as small bound as possible. **Solution:** *Since there are $|S|$ states,*

$$\mathit{Reach}(M) \subseteq \bigcup_{k=0}^{|S|} \bar{r}^k[I] \subseteq r_{\lceil \log |S| \rceil}[I]$$

In fact, $\lceil \log |S| \rceil$ is the smallest n such that $\bar{r}^k[I] \subseteq r_n[I]$ for every k such that $0 \leq k \leq 2^n$ $\Diamond$

Exercise 5 (Approximating Relations)**.** Consider a guarded command language whose meanings are binary relations on the set states U.

Let $E(a_1, \ldots, a_n)$ denote an expression built from some atomic relations $a_1, \ldots, a_n$, as well as diagonal relations

$$\Delta_P = \{(x, x) \mid x \in P\}$$

for various sets $P \subseteq U$. The expression E is built from these relations using union (to model non-deterministic choice, relation composition (to represent sequential composition) and transitive closure (to represent loops).

Let us call a relation $s \subseteq U \times U$ an *effect* if it is reflexive (R) and transitive (T).

A) (2pt) Prove that if s is an effect and $a_i \subseteq s$ for all $1 \leq i \leq n$, then

$$E(a_1, \ldots, a_n) \subseteq s$$

Solution: *By structural induction:*

Base case:

$a_i \subseteq s$ for all $1 \leq i \leq n$ and since s is reflexive, $\Delta_P \subseteq s$ for every $P \subseteq U$.

Induction:

If $E(a_1, ..., a_n) = E_1(a_1, ..., a_n) \cup E_2(a_1, ..., a_n)$

$$E(a_1, ..., a_n) = E_1(a_1, ..., a_n) \cup E_2(a_1, ..., a_n) \subseteq s \cup s = s \text{ (by IH)}$$

If $E(a_1, ..., a_n) = E_1(a_1, ..., a_n) \circ E_2(a_1, ..., a_n)$: since by IH $E_1(a_1, ..., a_n), E_2(a_1, ..., a_n) \subseteq s$, and s is transitive,

$$E(a_1, ..., a_n) = E_1(a_1, ..., a_n) \circ E_2(a_1, ..., a_n) \subseteq s$$

If $E(a_1, ..., a_n) = E_1(a_1, ..., a_n)^*$

$$E(a_1, ..., a_n) = E_1(a_1, ..., a_n)^* \subseteq s^* = s$$

Where the inclusion is due to IH and the last equality to the fact that s is both transitive and reflexive. $\Diamond$

B) (2pt) Let $U = \mathbb{Z}^2$ denote pairs of integers, denoted by integer variables x, y. Let s be a specification relation given by the formula:

$$s = \{((x, y), (x', y')) \mid y \geq 0 \rightarrow (x' \leq x \wedge y' \geq 0)\}$$

Show that s is an effect.

C) (3pt) For the effect s in the previous point, prove that $\rho(p) \subseteq s$ where p is the following program (the initial values of variables can be arbitrary):

```
1  while (y ≥ 0) {
2    x = x − y;
3    if (x % 2 == 0)
4      y = y / 2
5    else
6      y = 3∗y + 1
7  }
```

Notation $\rho(p)$ denotes the relation corresponding to the program p.
Solution: ***Reflexivity:***

For arbitrary $(x, y) \in \mathbb{Z}^2$

$$y \geq 0 \rightarrow (x \leq x \wedge y \geq 0) \equiv y \geq 0 \rightarrow y \geq 0 \equiv T$$

Which implies that $((x, y)(x, y)) \in s$ *and therefore that* s *is reflexive.*

Transitivity:

Let $((x_1, y_1), (x_2, y_2))$ *and* $((x_2, y_2), (x_3, y_3)) \in s$. *The two following formula holds:*

$$y_1 \geq 0 \rightarrow (x_2 \leq x_1 \wedge y_2 \geq 0) \quad y_2 \geq 0 \rightarrow (x_3 \leq x_2 \wedge y_3 \geq 0)$$

Let's show if the formula $y_1 \geq 0 \rightarrow (x_3 \leq x_1 \wedge y_3 \geq 0)$ *holds.*

Assume $y_1 \leq 0$*: by the first statement* $x_2 \leq x_1$ *and* $y_2 \geq 0$ *hold. Since* $y_2 \geq 0$*, by the second statement* $x_3 \leq x_2$ *and* $y_3 \geq 0$ *are also true. Therefore* $x_3 \leq x_1$ *and* y_3.

Since s *is reflexive and transitive, it is an effect.-* $\Diamond$