EPFL Formal Verification Course, Quiz 2019-12-12

Each subquestion is scored independently.

Wrong answers do not penalize other parts.

You are allowed to use every true statement from the lecture slides provided that you refer to the lecture in which it is stated.

1 Invariants (16 points)

```
import stainless.collection._

object EvenCount {
  def evenCountAcc[T](l: List[T], p: T ⇒ Boolean, acc: Boolean): Boolean = {
    l match {
      case Nil() ⇒ acc
      case Cons(x, xs) ⇒ evenCountAcc(xs, p, p(x) != acc)
    }
  }

  def evenCount[T](l: List[T], p: T ⇒ Boolean): Boolean = {
    evenCountAcc(l, p, true)
  }
}
```

Figure 1: EvenCount

Consider the function `evenCount` given in Figure 1.

Question 1: Explain what `evenCount` function computes
(1 point)

Answer: It checks whether the number of elements of list `l` that satisfy predicate `p` is even.

Question 2: Write a postcondition for the value `res` returned by `evenCount`. Your postcondition $P($`res`, `l.count`$($`p`$))$ must be expressed in terms of `res` and `l.count(p)` (the latter counts the number of elements `x` in `l` such that `p(x)` holds). You can use standard arithmetic and boolean operations.
(5 points)

Question 3: Write a postcondition for the value returned by the auxiliary function `evenCountAcc` to prove that `evenCount` computes what you expressed in questions 1 and 2.

Your postcondition $Q($`res`, `acc`, `l.count(p)`$)$ must be expressed in terms of `res`, `acc`, and `l.count(p)`.
(7 points)

Answer: The following Stainless file passes verification and describes P and Q.

```
import stainless.collection._
object EvenCount {
  def P(res: Boolean, c: BigInt): Boolean =
    res == (c % 2 == 0)

  def Q(res: Boolean, acc: Boolean, c: BigInt): Boolean =
    res == (acc == (c % 2 == 0))

  def evenCountAcc[T](l: List[T], p: T => Boolean, acc: Boolean): Boolean= {
    l match {
      case Nil() => acc
      case Cons(x, xs) => evenCountAcc(xs, p, p(x) != acc)
    }
  } ensuring (res => Q(res, acc, l.count(p)))

  def evenCount[T](l: List[T], p: T => Boolean): Boolean = {
    evenCountAcc(l, p, true)
  } ensuring (res => P(res, l.count(p)))
}
```

Question 4: Explain why:

1. If the postcondition Q holds for the call `evenCountAcc(l, p, true)`, in `evenCount`, then the postcondition P holds for `evenCount`, i.e. why

$$Q(\texttt{res}, \texttt{true}, \texttt{l.count(p)}) \Rightarrow P(\texttt{res}, \texttt{l.count(p)})$$

holds (with all variables being universally quantified).

2. The postcondition Q holds in the `Nil` branch of `evenCountAcc`, i.e. why

$$Q(\texttt{acc}, \texttt{acc}, \texttt{Nil().count(p)})$$

holds (with all variables being universally quantified).

3. The postcondition Q holds in the `Cons` branch of `evenCountAcc`, assuming the value returned by the recursive call `evenCountAcc(xs, p, p(x)` $\neq$ `acc)` respects the post-condition). Said otherwise, explain why:

$$Q(\texttt{res}, \texttt{p(x)} \neq \texttt{acc}, \texttt{xs.count(p)}) \Rightarrow Q(\texttt{res}, \texttt{acc}, \texttt{Cons(x,xs).count(p)})$$

holds (again with all variables being universally quantified).

(3 points)

Answer: (Reference explanation omitted here. Various high-level and short explanations were accepted.)

2 Iterating sp and wp (28 points)

This question is inspired by relationships in a transition system. Let S be a non-empty set of states and $r \subseteq S \times S$. As usual, define

- for every $P \subseteq S$, $sp(P, r) = \{x' \mid \exists x \in P.(x, x') \in r\}$
- for every $Q \subseteq S$, $wp(r, Q) = \{x \mid \forall x'.(x, x') \in r \to x' \in Q\}$

Let $C = 2^S = \{X \mid X \subseteq S\}$ be the set of all sets of states. Let also: $Init \subseteq S$ and $Good \subseteq S$. Define:

- $F : C \to C$ as $F(X) = Init \cup sp(X, r)$
- $B : C \to C$ as $B(Y) = Good \cap wp(r, Y)$
- $l = \bigcup_{i \geq 0} F^i(\emptyset)$
- $h = \bigcap_{i \geq 0} B^i(S)$

The ordering relation we consider is $\subseteq$ relation on C. For each of the following determine if the property holds or not. If it holds, prove it (If you do not know how to prove it or you run out of time, explain why you believe it to be true for partial points). If you believe the property to be false, give a counterexample (no need to prove that it is a counterexample).

Answer: We give a short informal clarification as to why the property holds, even though this was not required for the answer to be considered correct.

Generally, observe that F is precisely *post* from Lecture 1.

1. F is monotonic
   Answer: True. When X becomes a superset, so does $sp(X, r)$ and thus $Init \cup sp(X, r)$.

2. B is monotonic **Answer:** True. When Y becomes a superset, so does $wp(r, Y)$ as Q appears on the right-hand side of $\to$ in the definition of wp. Also $\cap$ is monotonic, so the entire expression becomes a superset.

3. l is the least fixed point of F. **Answer:** True. F is omega-continuous with respect to $\subseteq$ relation on S, with a proof a minor variation of "Union-Distributivity in case of Increasing Sequence" in Lecture 14.

4. h is the greatest fixed point of B. **Answer:** True. This can be shown directly using the $\supseteq$ instead of $\subseteq$ order, or as follows. From Lecture 13 we have that $wp(r, Q) = S \setminus sp(S \setminus Q, r^{-1})$ so
$$B(Y) = Good \cap S \setminus sp(S \setminus Y, r^{-1})$$

Define $B_1(Z) = S \setminus B(S \setminus Z)$ (thus $B(Y) = S \setminus B_1(S \setminus Y)$) and let $Bad = S \setminus Good$. Then

$$B_1(Z) = Bad \cup sp(Z, r^{-1})$$

which has the same form as for F (but with r^{-1} as relation and Bad playing the role of $Init$). Consider the sequence

$$\emptyset, B_1(\emptyset), \ldots, B_1^i(\emptyset), \ldots$$

which represents states that can be reached going from error states backwards. The elements of that sequence are complements of the sequence

$$S, B(S), \ldots, B^i(S), \ldots$$

which follows by induction on i and definition of B_1:

$$B_1^{i+1}(S) = B_1(B_1^i(\emptyset)) = \text{ (I.H.) } B_1(S \setminus B^i(S)) = \text{ (definition of } B_1\text{) } S \setminus B(B^i(S)) = S \setminus B^{i+1}(S)$$

The sequence of B_1 converges to least fixed point of B_1, so the sequence of B converges to the greatest fixed point of B. The result to which it converges is given using $\bigcup$ when the sequence is increasing and using $\bigcap$ when it is decreasing.

5. $l \subseteq h$. **Answer:** Not necessarily. In fact, this condition holds precisely when $Good$ is an invariant of a transition system with the initial set $Init$ and a transition relation r. l represents reachable states starting from $Init$ and making steps using relation r. On the other hand, h represents states that are guaranteed to stay inside the set $Good$. As a counterexample, let $S = \{0, 1\}$, $Init = Good = \{0\}$ and $r = \{(0, 1)\}$. Then $l = \{0, 1\}$. Computation shows that $h = \emptyset$ because $Good \cap wp(r, Good) = \{0\} \cap \{1\} = \emptyset$.

6. $l \subseteq Good$ implies $Init \subseteq h$ **Answer:** True. Both conditions are stating that reachable states remain inside $Good$, that is, that there is no finite sequence $s_0, s_1, \ldots, s_n$ such that $n \geq 0$, $s_0 \in Init$, $s_n \in Bad$ and, for all $0 \leq i < n$, $(s_i, s_{i+1}) \in r$:

$$\begin{array}{l} l \subseteq Good \\ \bigcup_{i \geq 0} F^i(\emptyset) \subseteq Good \\ \forall i \geq 0.\ F^i(\emptyset) \subseteq Good \\ \forall i \geq 0.\ F^i(\emptyset) \cap Bad = \emptyset \\ \forall i. \geq 0.\ r^i[Init] \cap Bad = \emptyset \end{array}$$

whereas

$$\begin{array}{l} Init \subseteq h \\ Init \subseteq \bigcap_{i \geq 0} B^i(S) \\ \forall i \geq 0.\ Init \subseteq B^i(S) \\ \forall i \geq 0.\ Init \subseteq (S \setminus B_1^i(\emptyset)) \\ \forall i \geq 0.\ Init \cap B_1^i(\emptyset) = \emptyset \\ \forall i \geq 0.\ Init \cap (r^{-1})^i [Bad] = \emptyset \end{array}$$

7. $\mathit{Init} \subseteq h$ implies $l \subseteq \mathit{Good}$ **Answer:** True. See the explanation for the previous case.

Each part carries 4 points.

3 Fixed Points in Lattices (23 points)

Consider a complete lattice with the set of elements L and the partial order $\sqsubseteq$. (Remember that in a complete lattice, every set $X \subseteq L$ has the least upper bound and the greatest lower bound.)

Let $G : L \to L$ be a monotonic function with respect to $\sqsubseteq$. Let

$$\mathit{Fix} = \{x \in L \mid G(x) = x\}$$

be the set of all fixed points. Let $x, y \in \mathit{Fix}$ be two fixed points. Prove or disprove:

1. $x \sqcup y \sqsubseteq G(x \sqcup y)$ (4 points) **Answer:** True. We have $x \sqsubseteq x \sqcup y$ (because $x \sqcup y$ is an upper bound on $\{x, y\}$). By monotonicity of G, then $G(x) \sqsubseteq G(x \sqcup y)$. Analogously we obtain $G(y) \sqsubseteq G(x \sqcup y)$. This means that $G(x \sqcup y)$ is one upper bound on the set $\{G(x), G(y)\}$. Thus, the least among the upper bounds is smaller than it:

$$G(x) \sqcup G(y) \quad \sqsubseteq \quad G(x \sqcup y)$$

We have $G(x) = x$ and $G(y) = y$ because $x, y \in \mathit{Fix}$, so the desired property holds.

2. $G(x \sqcup y) \sqsubseteq x \sqcup y$ (6 points) **Answer:** False. The property would hold if G was distributing over $\sqcup$, so we need to find an example where this is not the case. There are many counterexamples, but let us take inspiration from the counter-example to union-distributivity of contexts E in Lecture 14. Take L to be the set of all relations on the set $S = \{1, 2, 3\}$ with $\sqcup$ be $\cup$ and and define $G(r) = \Delta_S \cup (r \circ r)$. Then any reflective and transitive relation is a fixed point of G, including

$$r_1 = \{(1, 1), (1, 2), (2, 2)\}$$

$$r_2 = \{(2, 2), (2, 3), (3, 3)\}$$

We gave $r_1 \cup r_2 = \{(1, 1), (1, 2), (2, 2), (2, 3), (3, 3)\}$, which is not transitive. $G(r_1 \cup r_2)$ does contain $(1, 3)$, so the inclusion does not hold.

3. $x \sqcup y \in \mathit{Fix}$ (4 points) **Answer:** False. By definition, $x \sqcup y \in \mathit{Fix}$ means $G(x \sqcup y) = x \sqcup y$ and this implies $G(x \sqcup y) \sqsubseteq x \sqcup y$, which we have just shown to be false.

4. Let $B = \{b \in Fix \mid x \sqsubseteq b \wedge y \sqsubseteq b\}$. Then B has the least element, that is, an element $z \in B$ such that $\forall b \in B.\ z \sqsubseteq b$ (9 points. Possibly difficult.) **Answer:** True. The statement says that there exists the least upper bound, with respect to our existing ordering, in the set of fixed points. Note that this will not generally be $x \sqcup y$ but, by definition we will have $x \sqcup y \sqsubseteq z$ as $z \in B$. Inspired by the proof that the least fixed point of G is

$$\sqcap\{u \in L \mid G(u) \sqsubseteq u\}$$

let us consider a similar set restricted to elements above $x \sqcup y$ and define:

$$M = \{u \in L \mid x \sqcup y \sqsubseteq u \ \wedge \ G(u) \sqsubseteq u\}$$

We then repeat the proof for the version of Tarski's fixed point theorem that we stated at the beginning of Lecture 20, but using M instead of $Post$ as the set. Let $z = \sqcap M$. Take any $u \in M$. Then $z \sqsubseteq u$ as z is a lower bound. By monotonicity, $G(z) \sqsubseteq G(u) \sqsubseteq u$. Thus, $G(z)$ is a lower bound on M. Since z is the greatest of the lower bounds on M, we get $G(z) \sqsubseteq z$. By definition of M, it is also the case that $x \sqcup y$ is a lower bound on M, so $x \sqcup y \sqsubseteq z$. Thus, $z \in M$. By monotonicity $G(G(z)) \sqsubseteq G(z)$ and $G(x \sqcup y) \sqsubseteq G(z)$. By previous part, $x \sqcup y \sqsubseteq G(x \sqcup y) \sqsubseteq G(z)$. We thus get that also $G(z) \in M$. Since z is a lower bound on M, $z \sqsubseteq G(z)$. Thus, we have $G(z) = z$, so $z \in Fix$. So, everything worked similarly as in the original proof.

From $z \in M$ we have $x \sqsubseteq z$ and $y \sqsubseteq z$. We have also shown $z \in Fix$, so $z \in B$. We will show that $z \sqsubseteq b$ for all $b \in B$. Fix arbitrary $b \in B$. Then $x \sqcup y \sqsubseteq b$ and $G(b) = b$ so $G(b) \sqsubseteq b$. Thus also $b \in M$ (indeed, $B \subseteq M$). Since z is a lower bound on M, we have $z \sqsubseteq b$. Thus, we have shown that z is the least element of B.

4 Reasoning about Simple Integer Linear Inequalities (24 points)

Fix a set of variables V and let $\mathcal{F}$ be the set of conjunctions of literals of the form $x \leq y + c$ where $x, y \in V$ and where c is an arbitrary integer constant (possibly different for each literal):

$$\bigwedge_{i=1}^{k} x_{p_i} \leq x_{q_i} + c_i$$

For example, if $V = \{x, y, z\}$, an example $F \in \mathcal{F}$ is:

$$x \leq y + 1 \wedge y \leq z + (-2) \wedge z \leq x + 0$$

1. give an algorithm that, for $F \in \mathcal{F}$ and a variable $y \in V$, computes a formula $G \in \mathcal{F}$ that is equivalent to $\exists y.F$
(8 points) **Answer:** Without loss of generality we can assume that for every distinct

pair of variables x, y at most one conjuct $x \leq y + c$ appears because the conjunct with the smallest c implies all the others. If we have a constraint $x \leq x + c$ then for $c \geq 0$ it can be dropped and for $c < 0$ the entire formula is equivalent to false and we can represent it using one specific formula such as $x \leq x - 1$. Eliminating y from a formula that does not contain it results in the same formula. Eliminating y from $y \leq y - 1$ results in false formula. ($\mathcal{F}$ as defined does not explicitly have a way of representing False without using at least one variable. However, the task does not require that G should have a subset of variables of G, so using $x \leq x - 1$ for arbitrary x to represent $False$ is acceptable.)

Using analogous technique as in quantifier elimination for rational arithmetic, we eliminate the variable y by combining all of its lower bounds with all of its upper bounds. For every equation of the form $x \leq y + c_1$ for some x and c_1 and every equation $y \leq z + c_2$ for some z and c_2, we add the two inequations canceling y and obtain $x \leq z + c$ where $c = c_1 + c_2$. Doing this for all lower and upper bounds and conjoining all such inequalities results in a formula G (if there are no such pairs, we define the resulting formula to be the trivial formula $True$). G does not contain y and has no new variables either.

The resulting formula is equivalent to $\exists y.F$, though it was not required to explicitly prove this. Because each conjunct of G is a consequence of $\exists y.F$, G is also a consequence of $\exists y.F$ (we have $F \models G$). To show the converse, observe that $\exists y.F$ is equivalent to the maximum of lower bounds on y being less than equal to the minimum of upper bounds on y and observe that G is stating the equivalent condition, because $maxA \leq minB$ for finite sets A, B is equivalent to $\bigwedge_{a \in A, b \in B} a \leq b$.

2. describe an algorithm that for $F \in \mathcal{F}$ determines if F is satisfiable, i.e., there exist values of variables for which formula is true
(8 points) **Answer:** Repeatedly applying the algorithm from the previous point, we can maintain formula so that it is either a representation of True (no conjuncts), a representation of False (as $x \leq x - 1$ for some variable x), or as a set of conjuncts $x \leq y + c$ where x, y are distinct variables. The number of such conjuncts is at most quadratic in the number of variables, and there is no need to consider variables other than those in the input formula. We terminate when the formula is trivially true or trivially false after the number of steps bounded by the number of variables in the input formula.

3. describe an algorithm that for $F, G \in \mathcal{F}$ checks whether $F \rightarrow G$ is a valid formula
(8 points) **Answer:** Given any algorithm from the previous case, we check

$$F \rightarrow \bigwedge_{i=1}^{n} x \leq y + c$$

by checking satisfiability of

$$F \wedge \neg(x \leq y + c)$$

where $\neg(x \leq y+c)$ is equivalent to $x > y+c$, i.e., $x \geq y+c+1$ and can be transformed into $y \leq x + (-c-1)$).

Analyze the efficiency of your algorithms and make them polynomial if possible. **Answer:** The algorithm in part 1 is quadratic at worst because of the need to create combination of every lower bound and every upper bound. The algorithm in part 2 has cubic upper bound because there is no need for more than one literal per an ordered pair of two variable names, so the total number of literals remains quadratic and we eliminate each variable that appears in the formula. (A more efficient satisfiability check is to apply transitivity of inequalities and report that the formula is satisfiable if no contradiction is found.) An algorithm in part 3 reduces to as many calls of algorithm in part 3 as there are conjuncts in G (again, more efficient algorithms exist).

5 Composing Galois Connections (9 points)

Galois connection (named after Évariste Galois) is defined by two monotonic functions $\alpha : C \to A$ and $\gamma : A \to C$ between partial orders $\leq$ on C and $\sqsubseteq$ on A, such that

$$\forall c, a. \quad \alpha(c) \leq_A a \iff c \leq_C \gamma(a) \qquad (*)$$

Denote the fact that α, γ are monotonic and that $(*)$ holds by

$$C \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} A$$

Assume we have two Galois connections:

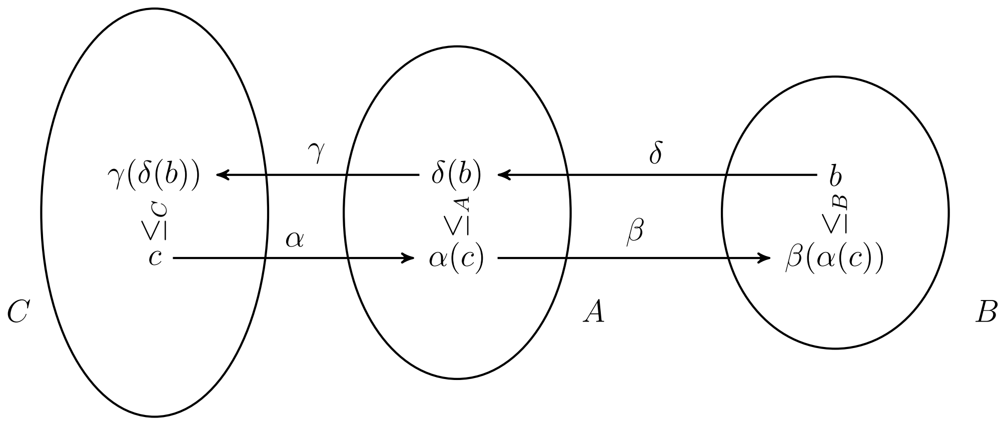
$$C \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} A \quad \text{and} \quad A \underset{\beta}{\overset{\delta}{\leftrightarrows}} B$$

Prove that $\beta \circ \alpha$ and $\gamma \circ \delta$ form a Galois connection between C and B:

$$C \underset{\beta\circ\alpha}{\overset{\gamma\circ\delta}{\leftrightarrows}} B$$

For any two functions, we define composition by $(f \circ g)(x) = f(g(x))$.

Here is a larger picture of the situation:



Each of the three sets C, A, B has its own ordering, but feel free to denote them all simply by $\leq$ if there is no danger that you confuse yourself.

Answer: Assume we have two Galois connections with monotonic $\alpha, \gamma, \beta, \delta$:

$$C \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} A \quad \text{and} \quad A \underset{\beta}{\overset{\delta}{\leftrightarrows}} B$$

As defined, to show

$$C \underset{\beta\circ\alpha}{\overset{\gamma\circ\delta}{\leftrightarrows}} B$$

we need to show that $\beta \circ \alpha$ as well as $\gamma \circ \delta$ are monotonic, as well as that the condition $(*)$ holds.

First, we show that composition of two monotonic functions is monotonic. Let α and β be monotonic and let $x \leq y$. By monotonicity of α, we have $\alpha(x) \leq \alpha(y)$. Now, by monotonicity of β, $\beta(\alpha(x)) \leq \beta(\alpha(y))$. By definition of function composition, the last condition is can be rewritten as $(\beta \circ \alpha)(x) \leq (\beta \circ \alpha)(y)$. This shows that $\beta \circ \alpha$ is monotonic.

The above proof also shows that $\gamma \circ \delta$ is monotonic.

It remains to show $(*)$ for $\beta \circ \alpha$ and $\gamma \circ \delta$. We use the fact that $(*)$ holds for α, γ as well as that it holds for β, δ:

$$\forall c, a. \quad \beta(a) \leq_B b \iff a \leq_A \delta(b) \quad (*')$$

We have the following set of equivalences:

$$\begin{aligned}
(\beta \circ \alpha)(c) \leq b &\iff && \text{(definition of composition)}\\
\beta(\alpha(c)) \leq b &\iff && (*')\\
\alpha(c) \leq \delta(b) &\iff && (*)\\
c \leq \gamma(\delta(b)) &\iff && \text{(definition of composition)}\\
c \leq (\gamma \circ \delta)(b) & &&
\end{aligned}$$