McMaster University
Department of Computing and Software
Musa Al-hassy

COMP SCI 3EA3 — Software Specification and Correctness

Feburary 7, 2017

Read Chapter 9 on "The Assignment Statement" and do the exercises in the text.

Exercise 6.0 — Division Algorithm and Ghost Variables

The division algorithm states, for any naturals n and $d \neq 0$ there are *unique* solutions q, r to $n = q \times d + r \ \wedge \ 0 \leq r < n$ and we call these two 'integer division' and 'remainder after division':

$$n \div d = q \ \wedge \ n \,\mathsf{mod}\, b = r \quad \equiv \quad n = q \times d + r \ \wedge \ 0 \leq r < n$$

This problem is devoted a whole chapter in the course text, chapter 15.

The idea that division is the number of times that the denominator goes into the numerator immediately gives rise to the program,

$$\begin{array}{l} q, r := 0, n, d \\ ; \ \textbf{do} \ r \geq n \to q, r := q + 1, r - b \ \textbf{od} \end{array}$$

However, this takes time $n \div d$ many steps to compute.

Numbers are implemented in binary and so we tend to have very fast division and multplication by 2 —by shifting the bits right and left. So assuming we have '$- \div 2$' as a primitive operation, we obtain a more efficient algorithm:

$$\begin{array}{l} q, r, b := 0, n, d \\ ; \ \textbf{do} \ r \geq b \to b := b * 2 \ \textbf{od} \\ ; \ \textbf{do} \ b \neq n \to \\ \qquad q, b := q * 2, b \div 2 \\ \qquad ; \textbf{if} \ r < b \to \mathsf{skip} \ [] \ r \geq b \to q, r := q + 1, r - b \ \textbf{fi} \\ \quad \textbf{od} \end{array}$$

This one essentially divides by $2 * d$ rather than just d and so takes half as many steps, and so twice as efficient! More formally, the variable b is *defined* by the invariant

$$n = q * b + r \ \wedge \ 0 \leq r < b \ \wedge \ \big(\exists k \ \bullet \ 0 \leq k \ \wedge \ b = 2^k * n\big)$$

Such an invariant may be difficult to work with due to the existential quantifier, what we can do is ignore the symbols '$\exists k \bullet$ ' and add "ghost variable" k and the aforementioned properties to our invariant; then we alter k in the program appropriately to maintain the invariant, such as `//@ ghost int k = 0; //@ ghost k = k + 1; //@ ghost k = k - 1;`

Implement both algorithms in C and prove them correct in `Frama-C`.

Exercise 6.1 — Pragrammatic Galois Connections, and other examples

Recall —see previous sheet!— the *Galois Connection* characterisation

$$L \dashv U \quad \equiv \quad (\forall x, y \bullet L\ x \sqsubseteq y \quad \equiv \quad x \sqsubseteq U\ y)$$

Prove the following are connections.

1. Negation is self-adjoint (in the dual order): $\neg p \Rightarrow q \quad \equiv \quad p \Leftarrow \neg q$ $\qquad (\neg) \dashv (\neg)$
2. Conjunction is adjoint with implication: $p \wedge q \Rightarrow r \quad \equiv \quad p \Rightarrow (q \Rightarrow r)$ $\qquad (\wedge q) \dashv (q \Rightarrow)$
   — this is essentially the division connection in a pomonoid!

   Since $p \Rightarrow q \quad \equiv \quad \neg p \vee q$, we can rephrase this connection as: $p \wedge q \Rightarrow r \quad \equiv \quad p \Rightarrow \neg q \vee r$ $\qquad (\wedge q) \dashv (\neg q \vee)$
   Using double negation, we obtain: $p \Rightarrow q \vee r \quad \equiv \quad p \wedge \neg q \Rightarrow r$ $\qquad (\wedge \neg q) \dashv (q \vee)$
3. Subtraction is self-adjoint (in the dual order): $-x \leq y \quad \equiv \quad x \geq -y$ $\qquad (-) \dashv (-)$
4. Addition is adjoint to subtraction: $x + y \leq z \quad \equiv \quad x \leq z - y$ $\qquad (+y) \dashv (-y)$

   More generally, every "order-isomorphism" pair form a Galois Connection

   $$f \fatsemi g = id = g \fatsemi f \quad \wedge \quad \mathsf{monotone}\ f \quad \wedge \quad \mathsf{monotone}\ g \quad \Rightarrow \quad f \dashv g$$

   for this reason Galois connections are known as "approximate inverses"!
5. Ceiling and floor are adjoint to real-number (implicit) type-casting, for $x : \mathbb{R}$ and $n : \mathbb{Z}$,

   $$\lceil x \rceil \leq n \quad \equiv \quad x \leq n \qquad \text{and, dually,} \qquad n \leq \lfloor x \rfloor \quad \equiv \quad n \leq x$$

   The first read right-to-left: "ceiling of x is the least integer containing x".
   The second read right-to-left: "floor of x is the *greatest* integer *contained in* x".
6. $$\mathsf{even}\ m \Leftarrow b \quad \equiv \quad \textbf{if}\ b\ \textbf{then}\ 2\ \textbf{else}\ 1\ \textbf{fi} \mid m \qquad \text{and, dually,} \qquad \mathsf{odd}\ m \Rightarrow b \quad \equiv \quad \textbf{if}\ b\ \textbf{then}\ 1\ \textbf{else}\ 2\ \textbf{fi} \mid m$$

   Similarly,

   $$x \in S \Leftarrow b \quad \equiv \quad S \supseteq \{e \mid e = x \quad \wedge \quad b\} \qquad \text{and, dually,} \qquad x \in S \Rightarrow b \quad \equiv \quad S \subseteq \{e \mid e = x \quad \Rightarrow \quad b\}$$

   Note that $\{e \mid e = x \quad \Rightarrow \quad b\} = \textbf{if}\ b\ \textbf{then U else U}\backslash\{x\}\ \textbf{fi}$ and $\{e \mid e = x \quad \wedge \quad b\} = \textbf{if}\ b\ \textbf{then}\ \{x\}\ \textbf{else}\ \varnothing\ \textbf{fi}$
   Moreover the property "lower adjoints can always look up" applied to these last two yield the famialr laws —verify! Note the dual order!—

   $$x \in (\cap i : 0..n \bullet S_i) \quad \equiv \quad (\forall i : 0..n \bullet x \in S_i) \qquad \text{and, dually,} \qquad x \in (\cup i : 0..n \bullet S_i) \quad \equiv \quad (\exists i : 0..n \bullet x \in S_i)$$

   We get these for *free* since we've shown membership is a lower adjoint in two ways!

Heuristic: to prove a function distributes over arbitrary meets or joins, try to show it is part of a Galois Connection

7. Direct image is adjoint with preimage: $f^{\rightarrow} A \subseteq B \quad \equiv \quad A \subseteq f^{\leftarrow} B$ $\qquad f^{\rightarrow} \dashv f^{\leftarrow}$

   where

   $$f^{\rightarrow} A = \{x \mid x \in A \bullet f\ x\} \qquad \text{and, dually,} \qquad f^{\leftarrow} B = \{x \mid f\ x \in B \bullet x\}$$

   The inverse image operation is interesting ;) Consider the operations '$\exists_f$' and '$\forall_f$' on sets

   $$\exists_f S = \{y \mid (\exists x \mid x \in f^{\leftarrow}\{y\} \bullet x \in S)\} \qquad \text{and, dually,} \qquad \forall_f S = \{y \mid (\forall x \mid x \in f^{\leftarrow}\{y\} \bullet x \in S)\}$$

   Then we have an adjoint triple: $\exists_f A \subseteq B \quad \equiv \quad A \subseteq f^{\leftarrow} B$ and $f^{\leftarrow} B \subseteq C \quad \equiv \quad B \subseteq \forall_f C$ $\qquad \exists_f \dashv f^{\leftarrow}$ and $f^{\leftarrow} \dashv \forall_f$
   Hence, quantifiers can be seen as adjoints in a Galois Connections!

   (Adjoints are unique and indeed it is not much trouble to show that $\exists_f = f^{\rightarrow}$)

   With these adjoints in-hand, the aforementioned heuristic, informs us that the direct image distributes over arbitrary unions, whereas the inverse image distributes over both arbitrary unions and arbitrary intersections —since it has both a lower and an upper adjoint!

8. Strongest transfomer is adjoint with the weakest: $\mathsf{sp}(S,p) \Rightarrow q \quad \equiv \quad p \Rightarrow \mathsf{wp}(S,q)$ $\qquad \mathsf{sp}\ S \dashv \mathsf{wp}\ S$

   where $\{p\}S\{q\}$ means that program S when executed in state p is guaranteed to terminate in a state satisfying q, and $\mathsf{sp}(S,p)$ is the strongest *post*condition candidate q while $\mathsf{wp}(S,q)$ is the weakest *pre*condition candidate p:

$$\{p\}S\{q\} \quad \equiv \quad p \Rightarrow \mathsf{wp}(S,q) \qquad \text{and, dually,} \qquad \{p\}S\{q\} \quad \equiv \quad \mathsf{sp}(S,p) \Rightarrow q$$

9. Supremum/join is adjoint with the constant function: $\sup f \sqsubseteq u \quad \equiv \quad f \mathrel{\dot\sqsubseteq} K\ u$ $\qquad \sup \dashv K$

   Infimum/meet is adjoint with the constant function: $l \sqsubseteq \inf f \quad \equiv \quad K\ l \mathrel{\dot\sqsubseteq} f$ $\qquad K \dashv \inf$

   In-particular, for the Boolean lattice, the first one becomes:

$$(\exists x \bullet P\ x) \Rightarrow q \quad \equiv \quad (\forall x \bullet P\ x \Rightarrow q)$$

   which in set notation becomes:

$$S \neq \varnothing \Rightarrow b \quad \equiv \quad S \subseteq \{e \mid b\} \qquad \text{and, dually,} \qquad S = \varnothing \Leftarrow b \quad \equiv \quad S \subseteq \{e \mid \neg b\}$$

   Note $\{e \mid q\} = \textbf{if } q \textbf{ then U else } \varnothing \textbf{ fi}$.

   For a space of size *two*, we can specialize these definitions to get the usual characterizations of *binary* meet and join

$$z \sqsubseteq x \quad \wedge \quad z \sqsubseteq y \quad \equiv \quad z \sqsubseteq x \sqcap y \qquad \text{and, dually,} \qquad x \sqsubseteq z \quad \wedge \quad y \sqsubseteq z \quad \equiv \quad x \sqcup y \sqsubseteq z$$

   In turn these can be specialized to the case of natural numbers ordered by division to obtain

$$k \mid m \quad \wedge \quad k \mid n \quad \equiv \quad k \mid x\ \mathsf{gcd}\ y \qquad \text{and, dually,} \qquad m \mid k \quad \wedge \quad n \mid k \quad \equiv \quad m\ \mathsf{lcm}\ n \mid k$$

   Finally, applying these to a space of size '*one*' —let $\mathbf{1} = \{\star\}$ be the set with one element named '$\star$'— then elements $\bot$ and $\top$ are the top and bottom elements of a poset precisely when $K\bot \dashv K\star$ and $K\star \dashv K\top$; **Verify!**

10. Composition is adjoint to residuation $\qquad (\fatsemi S) \dashv (/S)$ and $(R \fatsemi) \dashv (R\backslash)$

$$R \fatsemi S \subseteq T \quad \equiv \quad R \subseteq T/S \qquad \text{and, dually,} \qquad R \fatsemi S \subseteq T \quad \equiv \quad S \subseteq R\backslash T$$

    where "relation composition" and inclusion are defined by

$$a\ (R \fatsemi S)\ c \quad \equiv \quad (\exists b \bullet a\ R\ b \quad \wedge \quad b\ S\ c) \qquad \text{and} \qquad R \subseteq S \quad \equiv \quad (\forall x, y \bullet x\ R\ y \quad \Rightarrow \quad x\ S\ y)$$

    and residuals, or 'factors', are defined by

$$x\ (T/S)\ y \quad \equiv \quad (\forall z \bullet x\ T\ y \quad \Leftarrow \quad y\ S\ x) \qquad \text{and, dually,} \qquad x\ (R\backslash T)\ y \quad \equiv \quad (\forall z \bullet z\ R\ x \quad \Rightarrow \quad z\ T\ y)$$

    Notice that (1) the connections are nothing more than division in a pomonoid, and (2) the residual definitions *generalize* the principles of **indirect inclusion!**

11. Covariance and contravariance: If $L \dashv U$ then $(\fatsemi L) \dashv (\fatsemi U)$ and $(U \fatsemi) \dashv (L \fatsemi)$.

12. Pointwise meets are adjoint with pointwise joins
    if $L \dashv U$ and $L' \dashv U'$ then $(L \mathrel{\dot\sqcap} L')\ x \sqsubseteq y \quad \equiv \quad x \sqsubseteq (U \mathrel{\dot\sqcup} U')\ y$ $\qquad L \mathrel{\dot\sqcap} L' \dashv U \mathrel{\dot\sqcup} U'$

    where the pointwise extension of a binary operation $\oplus$ is defined by: for any functions f and g, $(f \mathrel{\dot\oplus} g)\ x = f\ x \oplus g\ x$. Exercises, similar to the setoids and posets, show that a monoid $(M, \oplus, e)$ gives rise to a "pointwise monoid" $(X \to M, \mathrel{\dot\oplus}, K\ e)$, for any type X.

13. **Many problems in computing are of the form "find a candidate to a problem, and make sure it is optimal in some way" and these can be *specified* using Galois Connections, from which implementations may be derived.**
    For example, "`take n xs` is the longest prefix of `xs` with at-most n items" yields connection:

$$ys \sqsubseteq \mathsf{take}\ n\ xs \quad \equiv \quad \mathsf{length}\ ys \leq n \quad \wedge \quad ys \sqsubseteq xs$$

where '$\sqsubseteq$' is the prefix relation on lists. A more informal definition of `take` is

$$\mathsf{take}\ n\ [x_0, x_1, \ldots, x_l] = [x_0, x_1, \ldots, x_n]$$

Using the former, the *specfication*, it may be easier to reason about this function, while using the latter, the *direct definition*, means we are forced to prove properties about it using induction —since it is defined directly as a recursive operation.

For example, **prove using the connection and then prove using induction**, and see how the latter is more cumbersome, the following properties

$$\mathsf{take}\ m \fatsemi \mathsf{take}\ n = \mathsf{take}\ (m \downarrow n) \qquad \text{and} \qquad \mathsf{take}\ n\ xs = xs \quad \equiv \quad \mathsf{length}\ xs \leq n$$

Notice that the connection, like pointwise order, lives in a product-space and as such can be seen more direcly as: $ys \sqsubseteq \mathsf{take}(n, xs) \quad \equiv \quad (\Delta \fatsemi (\mathsf{length} \times Id))\ ys\ (\leq \times \sqsubseteq)\ (n, xs)$ where
$\Delta\ y = (y, y)$, $(f \times g)(x, y) = (f\ x, g\ y)$, and $(a, b) \leq \times \sqsubseteq (c, d) \quad \equiv \quad a \leq c \wedge b \sqsubseteq c$.
These pieces allow also us to view binary joins and meets as Galois Connections.

Spefiy the following operations,

- The dual of `take`, also known as `drop` —use the suffix order.
- The self-explanatory function for the "longest common subsequence" —use the subsequence order.
- "`takeWhile p xs` is the longest prefix of `xs` whose elements all satisfy predicate `p`."

Exercise 6.2 — Program Derivation: Optional Assignment, 2% —due before the next quiz

Do the previous "Optional Assignment", at its original worth,

OR

for 2%, select an algorithm on lists —say from the Haskell prelude—, specfiy it as a Galois Connection and then use *that specification* to derive an implementation. Write and submit a report similar to the indications on Sheet 5.

An example of such a derivation —for the division algorithm of the previous sheet— can be found on page 73 of Principles and Applications of Algorithmic Problem Solving by João Ferreira.

Exercise 6.3 — Fixed Points

See §6.1 of Roland Backhouse's tutorial on motivating the need for fix points.

The purpose of this exercises is to prove

Knaster-Tarski Lemma: every monotonic function on a complete lattice has a least and greatest fixed point.

Assume,

$$f \text{ monotonic} \text{ —i.e., } (\forall x, y \bullet x \sqsubseteq y \;\Rightarrow\; f\ x \sqsubseteq f\ y)$$

and let

$$\mu f = (\sqcap x \mid f\ x \sqsubseteq x \bullet x)$$

This is our candidate for the least fixed point; let's prove that it is.

Prove —see the theorems list on the website!—

$$\mu f\text{-induction rule: } (\forall x \bullet \mu f \sqsubseteq x \;\Leftarrow\; f\ x \sqsubseteq x)$$

With that in-hand, complete the following proof:

Proving μf is a fixed point of f**:**

$\boxed{\ !\ }$

$=$ { antisymmetry }

$\boxed{\ !\ }$ $\wedge$ $\mu f \sqsubseteq f\ (\mu f)$

$=$ { let's work with the right conjunct first and see what we can do

$\mu f \sqsubseteq f\ (\mu f)$

$\Leftarrow$ { $\boxed{\ !\ }$ }

$f\ (\mu f)$ is a prefix point

$=$ { definition }

$f(f(\mu f)) \sqsubseteq f(\mu f)$

$\Leftarrow$ { $\boxed{\ !\ }$ }

$f\ (\mu f) \sqsubseteq \mu f$

Hence, we've proven $\mu f \sqsubseteq f\ (\mu f) \;\Leftarrow\; f\ (\mu f) \sqsubseteq \mu f$ now we can simplify the initial expression using "Definition of $\Leftarrow$": $p \Leftarrow q \;\equiv\; p \wedge q \;\equiv\; q$ }

$f(\mu f) \sqsubseteq \mu f$

$=$ { $\boxed{\ !\ }$ }

$f\ (\mu f) \sqsubseteq (\sqcap x \mid f\ x \sqsubseteq x \bullet x)$

$=$ { $\boxed{\ !\ }$ }

$\forall x \mid f\ x \sqsubseteq x \bullet f\ (\mu f) \sqsubseteq x$

$=$ { **Proving** $f\ (\mu f) \sqsubseteq x \;\Leftarrow\; f\ x \sqsubseteq x$**:**

$f\ (\mu f) \sqsubseteq x$

$\Leftarrow$ { $\boxed{\ !\ }$ }

$f\ (\mu f) \sqsubseteq f\ x \wedge f\ x \sqsubseteq x$

$\Leftarrow$ { $\boxed{\ !\ }$ }

$\mu f \sqsubseteq x \wedge f\ x \sqsubseteq x$

$=$ { $\boxed{\ !\ }$ }

$f\ x \sqsubseteq x$

}

true

Dualise the definition of μf to obtain 'νf' and prove that it is the *greatest* fixed point of f.

Do so in two ways: directly and via the duality principle.