McMaster University
Department of Computing and Software
Musa Al-hassy

COMP SCI 3EA3 — Software Specification and Correctness

January 31, 2017

Exercise 4.1 — Setoids

Recall that a *setoid*, or *E-set*, $(X, \sim)$ is a type X endowed with an equivalence relation $\sim$ —ie a relation that is reflexive, symmetric, and transitive —recall Sheet 2!

1. For any type X, show that $x \sim y \;:\equiv\; \mathit{true}$ is an equivalence relation.
2. For any setoid $(Y, \sim)$ and any function $f : X \to Y$, we have a "retract setoid": show that $(X, \sim_f)$ is a setoid where $a \sim_f b \;:\equiv\; f\ a \sim f\ b$. on X.
3. ★ What is the name of the rule when we consider the retract setoid using $(Y, =)$?
4. Why isn't $\sim$ an equivalence relation, when we define $x \sim y \;:\equiv\; \mathit{false}$?
5. Equality can be defined in two ways:
   (a) There is a "least" equivalence relation on any type and it is denoted '='. —recall lecture notes!
   (b) There is an equivalence relation '=' on any type satisfying Lebiniz Rule.

   Argue that these two formulations are identical.

Exercise 4.2 — Lattices

Recall that a lattice $(L, \sqsubseteq, \sqcap, \sqcup)$ is a *poset* $(L, \sqsubseteq)$ endowed with two operations 'meet' $\sqcap$ and 'join' $\sqcup$ which produce the *greatest lower bound* and *least upper bound*, respectively. The axioms for the operations can be succintly phrased as

$$\text{"meet characaterisation"}:\ z \sqsubseteq x \ \wedge\ z \sqsubseteq y \ \equiv\ z \sqsubseteq x \sqcap y$$

$$\text{"join characaterisation"}:\ x \sqsubseteq z \ \wedge\ y \sqsubseteq z \ \equiv\ x \sqcup y \sqsubseteq z$$

1. Prove that the lattice operations are: idempotent, associative, and symmetric. —review Sheet 2!
   Use the following principle and its dual

   $$\text{"indirect equality from above"}:\ l = r \ \equiv\ (\forall a \ \bullet\ l \sqsubseteq a \ \equiv\ r \sqsubseteq a)$$

2. Show that $(\mathbb{N}, \leq, \downarrow, \uparrow)$ is a lattice.
3. Show that $(\mathbb{P}X, \subseteq, \cap, \cup)$ is a lattice for any type X.
4. Show that $(\mathbb{N}, |, \mathsf{gcd}, \mathsf{lcm})$ is a lattice, where the ordering is by division: $x|y \;:\equiv\; (\exists k : \mathbb{N} \ \bullet\ y = k \times x)$
5. What is the "Golden Rule" in the previous lattice and how is it related to the formula

   $$(a\ \mathsf{gcd}\ b) \times (a\ \mathsf{lcm}\ b) = a \times b$$

   Is this a true formula? What about in the setting of first lattice above? —using min, max in-place of gcd, lcm.
6. The meet of the third lattice above can be computed by the following algorithm

   $$\textbf{do}\ a > b \to a := a - b \ \ [] \ \ a < b \to b := b - a\ \textbf{od}$$

   Formalise this algorithm in C and annotate it with assertions; you will need to use certain lattice facts about gcd —see first item above— and the property

   $$x < y \Rightarrow x\ \mathsf{gcd}\ y = x\ \mathsf{gcd}\ (y - x)$$

Exercise 4.3 — Categories

Recall that a *Category* $(Obj, \longrightarrow, \fatsemi, Id)$ consists of a type Obj of "objects" and for each pair of objects x, y it consists of a type $x \longrightarrow y$ of "morphisms" from x to y, and we have an inhabitant $Id_x : x \longrightarrow x$ which is the unit of the "composition" $\fatsemi$ which in-turn is an associative operation and has typing rule: if $f : x \longrightarrow y$ and $g : y \longrightarrow z$ then $(f \fatsemi g) : x \longrightarrow z$.

1. Show that Sets as objects with functions as morphisms constitute a category.
2. Argue that types in a programming language as objects with methods of that language as morphisms constitute a category.
3. ★ Argue that every category is a graph —we mearly ignore some structure.

   Motto: "categories are graphs endowed with algebraic structure"

   Somewhat conversely, show that the collection of paths on a graph consitute a category —with objects being the nodes of the graph.
4. ★ Argue that a category with only one object is essentially a *monoid*.

   Motto: "categories are typed monoids"

   Conversely, show that a monoid $(M, \oplus, e)$ gives rise to a category with one unnamed object and the morphims are the elements of the monoid, with composition being $\oplus$.
5. ★ Argue that a category having only one element for each morphism type is essentially a *poset*.

   Motto: "categories are constructive posets"

   Conversely, show that a poset $(P, \sqsubseteq)$ gives rise to a category $(P, \longrightarrow, \mathsf{trans}, \mathsf{refl})$ where $x \longrightarrow y$ is the type of proofs that $x \sqsubseteq y$, if any at all.
6. A setoid $(S, \sim)$ is a special kind of poset and so gives rise to a category. Show that in this category, every morphism f has an "inverse morphism": a morphism g with $f \fatsemi g = g \fatsemi g = Id$.

   These kinds of categories are known as *groupoids*.
   (Adding inverses to a monoid yields a structure known as a *group*)

   Show that one-object groupoids are essentially setoids. What 'motto' do we have here?

Exercise 4.5 — CalcCheck

Log onto Avenue and try proving the theorems and solving the puzzles and problems taken from last term's comp sci 2dm3 class with Professor Wolfram Kahl, whose tool CalcCheck we will use to obtain immediate feedback on our proofs.

"Avenue → Contents → Exercises (CS2DM3 Prof Kahl)"

1. **Read**: chapters 7 —*Calculational Logic: Part 2*— of the course text.
2. Add proofs for all theorems in "2DM3 Exercise 5.2-5.6".

Note that this exercises section is essentially tantamount to simply

"prove all theorems encountered when reading the course text"

however, using the tool increases **confidence in one's own proofs!**

Please send any feedback to tool provider at `kahl@cas.mcmaster.ca`!