Semantics of Programming Languages
Exercise Sheet 12

Exercise 12.1  Complete Lattices: GLB of UBs is LUB

Formalize the pen-and-paper proof from last homework (HW 11.1) as Isar-proof. Try to produce a proof whose structure is similar to the pen-and-paper proof.

**definition** "Sup' (S::'a::{complete_lattice, set}) ≡ Inf {u. ∀ s∈S. s≤u}"

**lemma** Sup'_upper: "∀ s∈S. s ≤ Sup' S"

**lemma** Sup'_least:
  **assumes** upper: "(∀ s∈S. s≤u)"
  **shows** "Sup' S ≤ u"

Exercise 12.2  Sign Analysis

Instantiate the abstract interpretation framework to a sign analysis over the lattice `pos, zero, neg, any`, where `pos` abstracts positive values, `zero` abstracts zero, `neg` abstracts negative values, and `any` abstracts any value.

For this exercise, you best modify the parity analysis `src/HOL/IMP/Abs_Int1_parity`

Homework 12.1  Small/Big Analysis

*Submission until Tuesday, 28. 1.2014, 10:00am.* Instantiate the abstract interpretation framework to find out which variables have values in the range \{-128...127\}, i.e. fit into one byte.

Start your development from `src/HOL/IMP/Abs_Int1_parity`. You do not need to show termination.

Homework 12.2  Kleene fixed point theorem

*Submission until Tuesday, 28. 1.2014, 10:00am.* Prove the Kleene fixed point theorem.

We first introduce some auxiliary definitions:
A chain is a set such that any two elements are comparable. For the purposes of the Kleene fixed-point theorem, it is sufficient to consider only countable chains. It is easiest to formalize these as ascending sequences. (We can obtain the corresponding set using the function `range :: (a ⇒ b) ⇒ b set`.)

**definition** chain :: “(nat ⇒ 'a::complete_lattice) ⇒ bool”

where “chain C ←→ (∀ n. C n ≤ C (Suc n))”

A function is continuous, if it commutes with least upper bounds of chains.

**definition** continuous :: “('a::complete_lattice ⇒ 'b::complete_lattice) ⇒ bool”

where “continuous f ←→ (∀ C. chain C −→ f (Sup (range C)) = Sup (f ' range C))”

The following lemma may be handy:

**lemma** continuousD: “[continuous f; chain C] −→ f (Sup (range C)) = Sup (f ' range C)”

**unfolding** continuous_def by auto

As warm-up, show that any continuous function is monotonic:

**lemma** cont_imp_mono:

fixes f :: “'a::complete_lattice ⇒ 'b::complete_lattice”

assumes “continuous f”

shows “mono f”

Hint: The relevant lemmas are

**thm** mono_def monoI monoD

Finally show the Kleene fixed point theorem. Note that this theorem is important, as it provides a way to compute least fixed points by iteration.

**theorem** kleene_lfp:

fixes f :: “'a::complete_lattice ⇒ 'a”

assumes CONT: “continuous f”

shows “lfp f = Sup (range (λ i. (fˆ^i) bot))”

**proof**

We propose a proof structure here, however, you may deviate from this and use your own proof structure:

let ?C = “λi. (fˆ^i) bot”

note MONO=cont_imp_mono[OF CONT]

have CHAIN: “chain ?C”

show ?thesis

proof (rule antisym)

show “Sup (range ?C) ≤ lfp f”

next

show “lfp f ≤ Sup (range ?C)”

qed

qed

Hint: Some relevant lemmas are

**thm** lfp_unfold lfp_lowerbound Sup_subset_mono range_eqI