Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Fabian Huch, Kevin Kappelmann

WS 23/24
23.01.2024

Semantics of Programming Languages

Exercise Sheet 12

Exercise 12.1 Complete Lattice over Lists

Show that lists of the same length – ordered point-wise – form a partial order if the element type is partially ordered. Partial orders are predefined as the type class *order*.

instantiation *list* :: (*order*) *order*

Define the infimum operation for a set of lists. The first parameter is the length of the result list.

definition *Inf_list* :: *"nat ⇒ ('a::complete_lattice) list set ⇒ 'a list"*

Show that your ordering and the infimum operation indeed form a complete lattice:

interpretation *list_complete_lattice*:
  Complete_Lattice "{xs. length xs = n}" "Inf_list n" **for** *n*

Exercise 12.2 Bekić's theorem

In the following, we want to prove that the least fixed-point of a monotone function on pairs can be separated into a series of least fixed points. On wikipedia, the theorem is described as follows:

Bekić's theorem (called the "bisection lemma" in his notes) is that the simultaneous least fixed point $\mu(x, y).(f, g)(x, y) = (x_0, y_0)$ can be separated into a series of least fixed points, in particular:

$x_0 = \mu x.f(x, \mu y.g(x, y))$

$y_0 = \mu y.g(x_0, y)$

Proof (Bekić):

$y_0 = g(x_0, y_0)$ *since it is the fixed point. Similarly* $x_0 = f(x_0, \mu y.g(x_0, y)) = f(x_0, y_0)$. *Hence* x_0, y_0 *is a fixed point of* (f, g). *Conversely, if there is a pre-fixed point* (x_1, y_1) *with* $(x_1, y_1) \geq (f(x_1, y_1), g(x_1, y_1))$, *then* $x_1 \geq f(x_1, \mu y.g(x_1, y) \geq x_0$ *and* $y_1 \geq \mu y.g(x_1, y) \geq \mu y.g(x_0, y) = y_0$; *hence* $(x_1, y_1) \geq (x_0, y_0)$ (x_0, y_0) *is the minimal fixed point.*

In the following, we want to formalize this pen-and-paper proof:

lemma

fixes f g **assumes** *mono*: *"mono* $(\lambda(x,y).(f,\ g)(x,y))$*"*
defines *"*$x_0 \equiv \mu\ x.\ f\ (x,\mu\ y.\ g\ (x,y))$*"*
defines *"*$y_0 \equiv \mu\ y.\ g\ (x_0,y)$*"*
shows *"*$\mu(x,y).\ (f,g)(x,y) = (x_0,y_0)$*"*

Homework 12.1 Collecting Semantics

Submission until Monday, Jan 29, 23:59pm.

Consider the following program in an extended version of IMP.

```
| x := 9 {A0};
| y := 3 {A1};
| {A2}
| WHILE x > 0
| DO {A3}
|   x := x div y; {A4}
|   y := y + 1;   {A5}
| {A6}
```

Compute the collecting semantics: Show how the annotations change with each application of the step function, until you reach a fix-point.

Write down all entries for each column. If some value did not change, use the ________ (8 underscores) term. Write the states down using sets of (x,y) tuples in explicit form, i.e. sets of the shape $\{(x1,\ y2),\ (x2,\ y2),\ ...\}$. The first annotation is pre-defined by us.

definition A_0 :: *"entry list"* **where** *"*$A_0 = [V\{\}, V\{(9,0)\}$, ________, ________, ________, ________, ________, ________, ________, ________, ________, ________, ________ $]$*"*

definition A_1 :: *"entry list"*
definition A_2 :: *"entry list"*
definition A_3 :: *"entry list"*
definition A_4 :: *"entry list"*
definition A_5 :: *"entry list"*
definition A_6 :: *"entry list"*

Homework 12.2 Galois Connections

Submission until Monday, Jan 29, 23:59pm.

In the lecture, we will derive a generic framework for abstract interpretation. An abstract interpreter over-approximates the semantics of a program. For this, it symbolically executes the program, using abstract values instead of concrete ones. To do so, the abstract interpreter requires an abstraction f_{abs} for each operation f of the core language.

To obtain a sound and terminating interpreter, each abstraction f_{abs} must satisfy some conditions. Particularly, each abstraction must be (1) sound with respect to f and (2) monotone with respect to its abstract domain.

The aim of this exercise is to show that each such abstraction is just an over-approximation of something called a *Galois connection*; or in other words: Galois connections are the best (over-)approximations.

To define Galois connections, we first need to introduce two other concepts:

1. Given two relations R, S and functions f, g, we say that f, g are *related from R to S* if all R-related inputs x, y are mapped to S-related outputs $f\ x$, $g\ y$. Formally, the function relator is defined as follows:
   $R \Rrightarrow S \equiv \{(f,\ g).\ \forall x\ y.\ (x,\ y) \in R \longrightarrow (f\ x,\ g\ y) \in S\}$
2. Second, given two relations R, S and a function f, we say that f is *monotone from R to S* if f is related to itself from R to S. Formally:
   $(R \Rrightarrow m\ S)\ f \equiv (f,\ f) \in R \Rrightarrow S$

Fix two relations L, R such that L and R are preorders on their respective domains. Further fix two functions l, r such that l is monotone from L to R and r is monotone from R to L. We say that L, R, l, r forms a *Galois connection*, written $(L \dashv R)\ l\ r$, if $(x,\ r\ y) \in L \longleftrightarrow (l\ x,\ y) \in R$ for all $x \in \mathit{Domain}\ L$ and $y \in \mathit{Domain}\ R$.

We call L the *left relation*, R the *right relation*, l the *left adjoint*, and r the *right adjoint* of the Galois connection.[1]

In the case of abstract interpretation, L constitutes the order on the concrete domain, R the order on the abstract domain, $l\ x$ the abstraction of a concrete value x, and $r\ y$ the concretisation of an abstract value y.[2]

Given a concretisation function r, an abstract value y is a *sound abstraction* of a concrete value x if $r\ y$ over-approximates x. We define this relation formally:

$x\ {}_{L}\lessapprox_{R\ r}\ y \equiv y \in \mathit{Domain}\ R \land (x,\ r\ y) \in L$

Prove that $l\ x$ is a sound abstraction for Galois connections:

lemma *galois_con_sound_selfI*:
  assumes *"$(L \dashv R)\ l\ r$"*
    and *"$x \in \mathit{Domain}\ L$"*
  shows *"$x\ {}_{L}\lessapprox_{R\ r}\ l\ x$"*

Now prove the dual result: $l\ (r\ y)$ is an under-approximation of y.

lemma *galois_con_counit_underapproxI*:
  assumes *"$(L \dashv R)\ l\ r$"*
    and *"$y \in \mathit{Domain}\ R$"*
  shows *"$(l\ (r\ y),\ y) \in R$"*

[1] Galois connections are adjoints between preorder categories; hence the terminology.
[2] In practice, L often denotes the identity relation.

Given a relation S, we say that x, y are S-equal if $(x,\ y) \in S$ and $(y,\ x) \in S$. We write $x \equiv_S y$ in such cases.

Prove that the concretisation of a value is equivalent to the conretisation of its abstracted concretisation:

lemma *galois_con_right_counit_rel_eqI*:
  assumes "$(L \dashv R)\ l\ r$"
    and "$y \in \mathit{Domain}\ R$"
  shows "$r\ (l\ (r\ y)) \equiv_L r\ y$"

Now prove the dual result:

lemma *galois_con_left_unit_rel_eqI*:
  assumes "$(L \dashv R)\ l\ r$"
    and "$x \in \mathit{Domain}\ L$"
  shows "$l\ (r\ (l\ x)) \equiv_R l\ x$"

Show that Galois connections are, essentially, uniquely determined by either of l or r. We say "essentially" because l and r are only L- and R-equivalent up to changes outside the domains of L, R (i.e. values that are not of interest to us).

You only have to prove the case for fixed right adjoints:

lemma *galois_con_unique_left*:
  assumes "$(L \dashv R)\ l\ r$" "$(L \dashv R)\ l'\ r$"
  and "$x \in \mathit{Domain}\ L$"
  shows "$l\ x \equiv_R l'\ x$"

Now prove the converse:

lemma *galois_con_if_unique_left_if_galois_con*:
  assumes "$(L \dashv R)\ l\ r$"
    and "$\bigwedge x.\ x \in \mathit{Domain}\ L \Longrightarrow l\ x \equiv_R l'\ x$"
  shows "$(L \dashv R)\ l'\ r$"

A canonical example of a Galois connection is the one between the reals and integers, using the ceiling function $\lceil\cdot\rceil$. Prove it:

lemma *real_int_galois*: "$(\mathit{rel\_of}\ (\le) \dashv \mathit{rel\_of}\ (\le))\ \mathit{ceiling}\ \mathit{real\_of\_int}$"

Galois connections have some good closure properties. Prove that they are closed under composition:

lemma *galois_compI*:
  assumes "$(L \dashv M)\ l1\ r1$"
    and "$(M \dashv R)\ l2\ r2$"
  shows "$(L \dashv R)\ (l2\ o\ l1)\ (r1\ o\ r2)$"

As we have shown, given a Galois connection between a concrete and an abstract domain, we can obtain sound abstractions for all concrete values. However, we not only need to abstract values but also functions.

We hence want to show that Galois connections are closed under function relators. Unfortunately, this is only true for monotone functions. To restrict the function relator to such functions, we introduce the reflexive relator:

$R^{\oplus} \equiv \{(x,\, y).\ (x,\, x) \in R \wedge (y,\, y) \in R \wedge (x,\, y) \in R\}$

The set of monotone functions from R to S can now be expressed as $(R \Rrightarrow S)^{\oplus}$. To define appropriate functions for the Galois connection between function relators, we introduce the function mapper:

$(f \rightarrow g)\ h \equiv g \circ h \circ f$

Finally, you can prove the closure property. The construction is as follows:

context
  fixes *L1 R1 l1 r1 L2 R2 l2 r2 L R l r*
  assumes *gal_cons*: *"(L1 ⊣ R1) l1 r1" "(L2 ⊣ R2) l2 r2"*
  defines *L_def* [*simp*]: *"L ≡ (L1 ⇛ L2)*$^{\oplus}$*"*
  and *R_def* [*simp*]: *"R ≡ (R1 ⇛ R2)*$^{\oplus}$*"*
  and *l_def* [*simp*]: *"l ≡ (r1 → l2)"*
  and *r_def* [*simp*]: *"r ≡ (l1 → r2)"*
begin

First prove the characteristic property of Galois connections:

lemma *refl_fun_rel_galois_propI1*:
    assumes *"g ∈ Domain R"*
      and *"(f, r g) ∈ L"*
  shows *"(l f, g) ∈ R"*
lemma *refl_fun_rel_galois_propI2*:
    assumes *"f ∈ Domain L"*
      and *"(l f, g) ∈ R"*
  shows *"(f, r g) ∈ L"*

Now prove the main theorem:

theorem *galois_refl_fun_relI*: *"(L ⊣ R) l r"*
end

Given a Galois connection $(L \dashv R)\ l\ r$ between a concrete and an abstract domain, we can *automatically* derive an $(R \Rrightarrow R)$-monotone abstraction $f_{abs} := (r \rightarrow l)\ f$ for each $(L \Rrightarrow L)$-monotone operation f of the core language, according to the theorem we just proved.

This works for any n-ary operator. Here are two concrete examples, where we let Isabelle fill in the desired terms for the unknowns *?L*, *?R*, *?l*, *?r*. You can see the synthesised instantiations by printing the theorem.

Note: *schematic_goal* is just a variant of *lemma* that allows us to leave some terms unknown.

schematic_goal *real_int_galois2*:
  "(?L ⊣ ?R) (?l :: (real ⇒ real) ⇒ int ⇒ int) ?r"
  by (*rule galois_refl_fun_relI real_int_galois*)+

schematic_goal *real_int_galois3*:
  "(?L ⊣ ?R) (?l :: (real ⇒ real ⇒ real) ⇒ int ⇒ int ⇒ int) ?r"
  by (*rule galois_refl_fun_relI real_int_galois*)+

print_statement *real_int_galois3*

As expected, the Galois connection simply maps each abstract value to its concrecte representation, applies the concrete operation, and then abstracts the result. Here is an example:

lemma *"(real_of_int → (real_of_int → ceiling)) (+) i1 i2 = ⌈real_of_int i1 + real_of_int i2⌉"*
  unfolding *fun_map_eq* **by** (*rule refl*)

The following tasks are bonus exercises (5 bonus points in total).

Of course, we also want to know whether this abstraction is sound, i.e. f_{abs} maps sound approximations to sound approximations. Prove it:

lemma *galois_fun_sound*:
  assumes *gal_cons*: *"$(L1 \dashv R1)\ l1\ r1$" "$(L2 \dashv R2)\ l2\ r2$"*
  and *f_mono*: *"$(L1 \Rrightarrow m\ L2)\ f$"*
  and *ysound*: *"$x\ {}_{L1}{\lessapprox}_{R1\ r1}\ y$"*
shows *"$f\ x\ {}_{L2}{\lessapprox}_{R2\ r2}\ (r1 \to l2)\ f\ y$"*

Finally, we show that our construction not only gives us a sound but also the "best" (i.e. smallest) abstraction. In other words, the Galois abstraction $f_{abs} := (r \to l)\ f$ of f is an under-approximation of every sound abstraction of f. More formally: Fix relations L, R and a concretisation r. Assume f' is a sound abstraction of f. We want to show that $((r \to l)\ f,\ f') \in R \Rrightarrow R$.

There is a small problem with this endeavour: we only know about the concretisation r but there is no abstraction l we can use to construct $(r \to l)\ f$. Luckily, if R admits lower bounds, we can construct a canonical abstraction:

$l\ x := \bigsqcap\{y \in Domain\ R.\ (x,\ r\ y) \in L\}$.

context
  fixes *L* :: *"'a rel"* **and** *R* :: *"'b rel"* **and** *l r*
  and *lbound* :: *"'b set ⇒ 'b"*
  assumes *lbound_lower*: *"$\bigwedge Y\ y.\ Y \subseteq Domain\ R \Longrightarrow y \in Y \Longrightarrow (lbound\ Y,\ y) \in R$"*
  defines [*simp*]: *"$l \equiv \lambda x.\ lbound\ \{y \in Domain\ R.\ (x,\ r\ y) \in L\}$"*
begin

Now prove optimality of $(r \to l)\ f$. The following assumptions are sufficient:

lemma *galois_fun_optimal*:
    assumes *refl_onR*: *"refl_on (Domain R) R"*
      and *rmono*: *"$(R \Rrightarrow m\ L)\ r$"*
      and *f'sound*: *"$\bigwedge x\ y.\ x\ {}_{L}{\lessapprox}_{R\ r}\ y \Longrightarrow f\ x\ {}_{L}{\lessapprox}_{R\ r}\ f'\ y$"*
      and *f'mono*: *"$(R \Rrightarrow m\ R)\ f'$"*
  shows *"$((r \to l)\ f,\ f') \in R \Rrightarrow R$"*

Finally show that L, R, l, r indeed form a Galois connection provided that R admits infima and there is some Galois connection for L, R, r at all.

lemma *galois_fun_canonical*:

assumes *galcon*: “$(L \dashv R)\ l'\ r$”
and *lbound_least*: “$\bigwedge Y\ y'.\ Y \subseteq \mathit{Domain}\ R \Longrightarrow (\bigwedge y.\ y \in Y \Longrightarrow (y',\ y) \in R) \Longrightarrow (y',$
$\mathit{lbound}\ Y) \in R$”
shows “$(L \dashv R)\ l\ r$”