Technische Universität München
Fakultät für Informatik
Prof. Tobias Nipkow, Ph.D.
Simon Wimmer

WS 2019/20
27.01.2020

Semantics of Programming Languages

Exercise Sheet 12

Exercise 12.1 Inverse Analysis

Consider a simple sign analysis based on this abstract domain:

datatype *sign* = *None* | *Neg* | *Pos0* | *Any*

fun γ :: *"sign ⇒ val set"* **where**
"γ None = {}" |
"γ Neg = {i. i < 0}" |
"γ Pos0 = {i. i ≥ 0}" |
"γ Any = UNIV"

Define inverse analyses for "+" and "<" and prove the required correctness properties:

fun *inv_plus'* :: *"sign ⇒ sign ⇒ sign ⇒ sign * sign"*
lemma
"⟦ inv_plus' a a1 a2 = (a1',a2'); i1 ∈ γ a1; i2 ∈ γ a2; i1+i2 ∈ γ a ⟧
⟹ i1 ∈ γ a1' ∧ i2 ∈ γ a2' "
fun *inv_less'* :: *"bool ⇒ sign ⇒ sign ⇒ sign * sign"*
lemma
"⟦ inv_less' bv a1 a2 = (a1',a2'); i1 ∈ γ a1; i2 ∈ γ a2; (i1<i2) = bv ⟧
⟹ i1 ∈ γ a1' ∧ i2 ∈ γ a2'"

Exercise 12.2 Command Equivalence

Recall the notion of *command equivalence*:

$c_1 \sim c_2 \equiv (\forall s\ t.\ (c_1,\ s) \Rightarrow t \ \longleftrightarrow\ (c_2,\ s) \Rightarrow t)$

1. Define a function *is_SKIP* :: *com* ⇒ *bool* which holds on commands equivalent to *SKIP*. The function *is_SKIP* should be as precise as possible, but it should not analyse arithmetic or boolean expressions.

   Prove: *is_SKIP c* ⟹ $c \sim$ *SKIP*

2. The following command equivalence is wrong. Give a counterexample in the form of concrete instances for b_1, b_2, c_1, c_2, and a state s.

$$\begin{array}{l} \mathit{WHILE}\ b_1\ \mathit{DO}\ \mathit{IF}\ b_2\ \mathit{THEN}\ c_1\ \mathit{ELSE}\ c_2 \\ \sim\ \mathit{IF}\ b_2\ \mathit{THEN}\ (\mathit{WHILE}\ b_1\ \mathit{DO}\ c_1)\ \mathit{ELSE}\ (\mathit{WHILE}\ b_1\ \mathit{DO}\ c_2) \end{array} \quad (*)$$

3. Define a condition P on b_1, b_2, c_1, and c_2 such that the previous statement $(*)$ holds, i.e. $P\ b_1\ b_2\ c_1\ c_2 \Longrightarrow (*)$

   Your condition should be as precise as possible, but only using:

   - $\mathit{lvars} :: \mathit{com} \Rightarrow \mathit{vname\ set}$ (all left variables, i.e. written variables),
   - $\mathit{rvars} :: \mathit{com} \Rightarrow \mathit{vname\ set}$ (all right variables, i.e. all read variables),
   - $\mathit{vars} :: \mathit{bexp} \Rightarrow \mathit{vname\ set}$ (all variables in a condition), and
   - boolean connectives and set operations

General homework instructions

Both homework exercises are pen & paper (or keyboard & text file). You have the choice of uploading a textual solution in a theory file to the submission system or you can submit a PDF scan via E-Mail. Physical paper submissions are not accepted.

Homework 12.1 Intervals and Multiplication

Submission until Monday, February 3, 2020, 10:00am. (Pen & Paper)
Suppose we extend the arithmetic expression language of IMP with a multiplication construct. Explain how the abstract interpreter could be extended to handle multiplication on the interval domain:

1. Give the necessary additional definitions
2. Specify the newly arising proof obligations
3. Give a brief sketch for these proofs

You do not need to consider backward analysis of Boolean expressions and termination.

Homework 12.2 Complete Lattices

Submission until Monday, February 3, 2020, 10:00am. (Pen & Paper)

Let $'a$ be a complete lattice with ordering $\leq$ and let $f :: 'a \Rightarrow 'a$ be a monotone function.

Give a detailed proof for the following statement:

$$\bigsqcup\ (f\ `\ X) \leq f\ (\bigsqcup\ X)$$

Hints:

- The least upper bound satisfies the following properties
  - Upper bound: $x \in A \Longrightarrow x \leq \bigsqcup\ A$
  - Least upper bound: $(\forall\, x \in A.\ x \leq u) \Longrightarrow \bigsqcup A \leq u$
- $f\ `\ X$ is the function image: $f\ `\ X = \{y.\ \exists\, x{\in}X.\ f\, x = y\}$