Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Peter Lammich, Johannes Hölzl

WS 2013/14
28. 1. 2014

Semantics of Programming Languages

Exercise Sheet 13

Exercise 13.1 AI for Conditionals

Our current constant analysis does not regard conditionals. For example, it cannot figure out, that after executing the program

x:=*2*; *IF x<2 THEN x*:=*2 ELSE x*:=*1*

x will be constant.

In this exercise we extend our abstract interpreter with a simple analysis of boolean expressions. To this end, modify locale *Val_abs* in theory *Abs_Int0.thy* as follows:

- Introduce an abstract domain *'bv* for boolean values, add, analogously to *num'* and *plus'* also functions for the boolean operations and for *less*.
- Modify *Abs_Int0* to accomodate for your changes. Do not modify the locales ending in *_fun*, they are not needed for executable analysis, and can simply be commented out.
- Define a function *bval'* in *Abs_Int1*, and modify the *step'* function to take into account boolean values guaranteed to be false.
- Finally, adapt all theories necessary to get a more precise constant analysis.

Hint: Start with a fresh copy of the IMP/ folder.

Homework 13 Inverse Analysis

Submission until Tuesday, 4. 2 2014, 10:00am. 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′"