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

WS 2021/22
25.01.

Semantics of Programming Lectures

Exercise Sheet 12

Exercise 12.1 Complete Lattices

Which of the following ordered sets are complete lattices?

- $\mathbb{N}$, the set of natural numbers $\{0, 1, 2, 3, \ldots\}$ with the usual order
- $\mathbb{N} \cup \{\infty\}$, the set of natural numbers plus infinity, with the usual order and $n < \infty$ for all $n \in \mathbb{N}$.
- A finite set A with a total order $\leq$ on it.

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.

```
datatype sign = Pos | Zero | Neg | Any

instantiation sign :: order
instantiation sign :: semilattice_sup_top
fun γ_sign :: "sign ⇒ val set"
fun num_sign :: "val ⇒ sign"
fun plus_sign :: "sign ⇒ sign ⇒ sign"
global_interpretation Val_semilattice
  where γ = γ_sign and num' = num_sign and plus' = plus_sign
global_interpretation Abs_Int
  where γ = γ_sign and num' = num_sign and plus' = plus_sign
  defines aval_sign = aval' and step_sign = step' and AI_sign = AI
```

Some tests:

```
definition "test1_sign =
  ''x'' ::= N 1;;
  WHILE Less (V ''x'') (N 100) DO ''x'' ::= Plus (V ''x'') (N 2)"
value "show_acom (the(AI_sign test1_sign))"

definition "test2_sign =
  ''x'' ::= N 1;;
```

WHILE Less (V ''x'') (N 100) DO ''x'' ::= Plus (V ''x'') (N 3)"

definition *"steps c i = ((step_sign ⊤) ^^ i) (bot c)"*

value *"show_acom (steps test2_sign 0)"*

...

value *"show_acom (steps test2_sign 6)"*
value *"show_acom (the(AI_sign test2_sign))"*

Exercise 12.3 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_semilattice* 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 accommodate for your changes.

Homework 12.1 Prefix Analysis

Submission until Sunday, Jan 30, 23:59pm. In this homework, you shall modify IMP to work on strings rather than integers, and define an abstract interpretation based analysis to return all possible one-element prefixes of the values of a variable.

The string operations shall be:

- *Sc*: A string constant.
- *Conc*: Concatenate two strings.
- *Last*: Return the last character of the operand as a string of length one. Empty string if operand is empty.
- *Butlast*: Return the operand with the last character removed. Empty string if operand is empty.
- *Eq*: Compare two strings for equality.

otherwise keep the arithmetic and Boolean operations, i.e. Variables, And, Not.

The provided template file contains all necessary theory up to abstract interpretation in one file, with irrelevant parts stripped away. Use this template file as a basis for your modifications.

Finally, implement an abstract interpretation with the abstract domain *char option set*, which describes the set of possible one-element prefixes of a string, where *None* represents the empty string. Try to define the abstract operations as precise as possible! As starting point for your development, you can use the parity analysis, which is also included in the template file. You are required to interpret the locales *Val_semilattice*, *Abs_Int*, *Abs_Int_mono*, and *Abs_Int_measure*. Follow the naming schema, using ”prefixes” instead of ”parity”.