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

WS 2015/16
1. 12. 2015

Semantics of Programming Languages

Exercise Sheet 8

Exercise 8.1 Security type system: bottom-up with subsumption

Use the template file `ex08_tmpl.thy`.

Recall security type systems for information flow control from the lecture. Such a type systems can either be defined in a top-down or in a bottom-up manner. Independently of this choice, the type system may or may not contain a subsumption rule (also called anti-monotonicity in the lecture). The lecture discussed already all but one combination: a bottom-up type system with subsumption.

1. Define a bottom-up security type system for information flow control with subsumption rule.
2. Prove the equivalence of the newly introduced bottom-up type system with the bottom-up type system without subsumption rule from the lecture.

Exercise 8.2 Definite Initialization Analysis

Use the template file `ex08_tmpl.thy`.

In the lecture, you have seen a definite initialization analysis that was based on the big-step semantics. Definite initialization analysis can also be based on a small-step semantics. Furthermore, the ternary predicate D from the lecture can be split into two parts: a function $AA :: com \Rightarrow name\ set$ ("assigned after") which collects the names of all variables assigned by a command and a binary predicate $D :: name\ set \Rightarrow com \Rightarrow bool$ which checks that a command accesses only previously assigned variables. Conceptually, the ternary predicate from the lecture (call it D_{lec}) and the two-step approach should relate by the equivalence $D\ V\ c \longleftrightarrow D_{lec}\ V\ c\ (V \cup AA\ c)$

1. Study the already defined small-step semantics for definite analysis.
2. Define the function AA which computes the set of variables assigned after execution of a command. Furthermore, define the predicate D which checks if a command accesses only assigned variables, assuming the variables in the argument set are already assigned.
3. Prove progress and preservation of D with respect to the small-step semantics, and conclude soundness of D. You may use (and then need to prove) the lemmas *D_incr* and *D_mono*.

Homework 8 Definite Initialization II

Submission until Tuesday, Dec 8, 10:00am.

A well-initialized command only depends on the variables that are already initialized. That is, executability and the values of the definitely initialized variables after executing the command only depend on the values of the already initialized variables before the command.

Prove the following lemma, which formalizes the proposition above wrt. the standard big-step semantics. Import ~~/*src/HOL/IMP/Def_Init* and ~~/*src/HOL/IMP/Big_Step* for this homework.

lemma
  assumes *"D A c B"*
  assumes *"s1 = s2 on A"*
  assumes *"(c,s1) ⇒ s1'"*
  shows *"∃s2'. (c,s2) ⇒ s2' ∧ s1'=s2' on B"*