Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Sascha Böhme, Alexander Krauss

WS 2010/11
10. 01. 2011

Semantics of Programming Languages

Exercise Sheet 10

Exercise 10.1 Security type system: bottom-up with subsumption

Use the template file `Ex10_Template.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.

(a) Define a bottom-up security type system for information flow control with subsumption rule.

(b) Prove the equivalence of the newly introduced bottom-up type system with the bottom-up type system without subsumption rule from the lecture.

Homework 10

Submission until Wednesday, January 26, 2011, 12:00 (noon). The first part (the definition) is already due on January 19.

Use the template file `Dependency_Template.thy`.

The task is to define a dependency analysis between variables. We say that variable x depends on y after command c if the value of y at the beginning of the execution of c may influence the value of x at the end of the execution.

For example, consider the program $y ::= 0$; *IF* $x \leq 2$ *THEN* $y ::= x$ *ELSE* $z ::= 0$.

Here, the variable x depends only on itself, since it is never assigned.

The variable y clearly depends on x. It does not depend on itself, since it is initially assigned a constant value, hence the original value is irrelevant.

The variable z depends on itself, since it may keep its value, but it also depends on x, since the assignment to it occurs under a conditional depending on x.

In the program *WHILE* b *DO* ($x ::= y$; $y ::= z$) the variable x depends on both y and z (the value of z reaches x in the second iteration of the loop).

(a) Define an inductive relation *influences* :: *name* ⇒ *com* ⇒ *name* ⇒ *bool* which specifies a dependency analysis.

(b) Prove its soundness w.r.t. to the big-step semantics. That is, prove the lemma

lemma *deps_sound*:
"$[\![\ (c,\ s) \Rightarrow t;\ s = s'\ \mathit{on\ deps}\ c\ x;\ (c,\ s') \Rightarrow t'\ ]\!]$
$\Longrightarrow t\ x = t'\ x$"

where *deps c x* abbreviates $\{y.\ \mathit{influences}\ y\ c\ x\}$.