IST 597A HW3 Problem2 and Problem3

Yufei Jiang

1 Problem 2

1.1 Adding a type "ref τ" to Mini-C syntax

Types τ ::= bool | int | ref τ
Expressions e ::= x | v | $\odot$e | $e_1 \bigoplus e_2$ | γ x
Values v ::= n | true | false
ref Op γ ::= *
ref Values ψ ::= n | null

1.2 Typing Rules (Static Semantics)

$$\frac{\Gamma \vdash x : \text{ref } \tau}{\Gamma \vdash *x : \tau} \text{ TyExpDeref}$$

1.3 Small Step Rules

$$\frac{}{< \Delta, (x : \text{ref } \tau; D, s) > \vdash < \Delta[(x, \text{null})], (D, s) >} \text{ EvalProgDeclRef}$$

$$\frac{\Delta \rhd x : \text{ref } \tau}{< \Delta, x = \psi > \vdash < \Delta[(x, \psi)], \$ >} \text{ EvalStmAssignRefVal}$$

$$\frac{\Delta \rhd x \mapsto x'}{< \Delta, y = *x > \vdash < \Delta, y = *x' >} \text{ EvalStmDeref}$$

1.4 Proof of Type Safety

1.4.1 Preservation

Preservation assures that "if a well typed term takes one evaluation step, then the resulting term is again well-typed [1]". More specifically, "if e is a well-typed term, that is , $e : \tau$ for some type τ, and $e \to e'$, then $e' : \tau$ [1]" Our new added typing rules *TyExpDeref* and small step rule *EvalStmDeref* guaranteed that the property of preservation is retained. Specifically, given $x : \text{ref } \tau, y : \tau$, we know that $*x : \tau$. Also, $y = *x'$, thus, $x' : \text{ref } \tau$. So we always have $x \mapsto x'$, $x : \text{ref } \tau$ and $x' : \text{ref } \tau$ at the same time. Preservation is proved.

1.4.2 Progress

Progress means "to ensure that a well-typed term is not stuck (either it is a value or it can take step according to the evaluation rules) [1]". Specifically, "if e is a well-typed term, $e : \tau$ for some τ, then either e is a value, *eval*, or there exists e' such that $e \mapsto e'$ [1]"

From our rules and given e is well-typed, we can see that, e must be bool, int, or ref τ. Apparently, if the type of e is bool or int, then e is a value. If the type of e is ref τ, then is must be either null or the value of the reference. Here we implicitly use integer n to represent the value of the reference. Then it is still is a concrete value. Thus, the progress is proved.

Because the preservations and progress are proved, the type safety is proved.

2 Problem 3

2.1 Syntax

TypesWithQualifier $\mu ::= \xi\ \tau \mid \tau$
Qualifier $\xi ::=$ red

2.2 Proof of Type Safety

From the syntax definition, we can see that adding a security type qualifier does not import any new type. The definition of "Types" does not change at all. Since types are not changed, typing rules are not going to be changed. We need to define qualifier propagation rules which are out of the scope of typing rules. Original type safety is retained.

References

[1] A. Rentschler, *Model Transformation Languages with Modular Information Hiding*. KIT Scientific Publishing, 2015, vol. 17.