Type Soundness for TAL-0

Software Security

Spring 2025

The material in these notes has been adapted from Section 4 of [1].

The Goal

The goal of this lecture is to explain how we might prove that that any well-typed (i.e., derivable) machine state (H, R, I) cannot "get stuck". It suffices to show that for any machine state (H, R, I) that is well-typed (in the sense that $\vdash (H, R, I)$ is derivable in the TAL-0 type system) there is a machine state (H', R', I') such that $(H, R, I) \to (H', R', I')$ according to the operational semantics and that (H', R', I') well-typed.

The Argument

Suppose $\vdash (H, R, I)$ is well-typed. Then its derivation must end with the S-MACH inference rule, so we may conclude that for some heap type ψ and register valuation type Γ we have derivations of:

$$\vdash H : \psi \qquad \psi \vdash R : \Gamma \qquad \psi \vdash I : \mathsf{code}(\Gamma)$$

We proceed by structural induction on I, which must be one of:

- $\mathsf{jump}\ v$, in which case our derivation of $\psi \vdash I : \mathsf{code}(\Gamma)$ must end with the S-JUMP inference rule, which means that we have a derivation of:

  $$\psi; \Gamma \vdash v : \mathsf{code}(\Gamma)$$

  But this means that $\widehat{H}(\widehat{R}(v)) = I'$ for some instruction sequence I' for which $\psi \vdash I' : \mathsf{code}(\Gamma)$ is derivable. Now, the the JUMP rule gives $(H, R, \mathsf{jump}\ v) \to (H, R, I')$, and further we can derive $\vdash (H, R, I')$ using the S-MACH rule.

- $r_d := v; I'$, in which case our derivation of $\psi \vdash I : \mathsf{code}(\Gamma)$ must end with the S-SEQ inference rule, which means that for some Γ_2 we have derivations of:

  $$\psi \vdash r_d := v : \Gamma \to \Gamma_2 \qquad \psi \vdash I' : \mathsf{code}(\Gamma_2)$$

Now, our derivation of $\psi \vdash r_d := v : \Gamma \to \Gamma_2$ must end with the S-MOVE rule which means that we have a derivation of:

$$\psi; \Gamma \vdash v : \tau$$

and that $\Gamma_2 = \Gamma[(d, \tau)]$. The MOVE rule gives $(H, R, r_d := v; I') \to (H, R[(d, v)], I')$ and since obviously $\psi \vdash R[(d, v)] : \Gamma[(d, \tau)] = \Gamma_2$ we can derive $\vdash (H, R[(d, v)], I')$ using the S-SEQ rule.

- $r_d := r_s + v; I'$, in which case our derivation of $\psi \vdash I : \mathsf{code}(\Gamma)$ must end with the S-SEQ inference rule, which means that for some Γ_2 we have derivations of:

$$\psi \vdash r_d := r_s + v : \Gamma \to \Gamma_2 \qquad\qquad \psi \vdash I' : \mathsf{code}(\Gamma_2)$$

Now, our derivation of $\psi \vdash r_d := r_s + v : \Gamma \to \Gamma_2$ must end with the S-ADD rule, which means that we have derivations of:

$$\psi; \Gamma \vdash r_s : \mathsf{int} \qquad\qquad \psi; \Gamma \vdash v : \mathsf{int}$$

and that $\Gamma_2 = \Gamma[(d, \mathsf{int})]$. The existence of such derivations implies that $R(s) = n_1 \in \mathbb{Z}$ and $\widehat{R}(v) = n_2 \in \mathbb{Z}$, which means that the ADD rule gives $(H, R, r_d := r_s + v; I') \to (H, R[(d, n_1 + n_2)], I')$. Obviously we have $\psi \vdash R[(d, n_1 + n_2)] : \Gamma[(d, \mathsf{int})]$, and so we can derive $\vdash (H, R[(d, n_1 + n_2)], I')$ using the S-SEQ rule.

- $\mathsf{if}\ r_i\ \mathsf{jump}\ v; I'$, in which case our derivation of $\psi \vdash I : \mathsf{code}(\Gamma)$ must end with the S-SEQ inference rule, which means that for some Γ_2 we have derivations of:

$$\psi \vdash \mathsf{if}\ r_i\ \mathsf{jump}\ v : \Gamma \to \Gamma_2 \qquad\qquad \psi \vdash I' : \mathsf{code}(\Gamma_2)$$

Now, our derivation of $\psi \vdash \mathsf{if}\ r_i\ \mathsf{jump}\ v : \Gamma \to \Gamma_2$ must end with the S-COND inference rule, which means that we have derivations of:

$$\psi; \Gamma \vdash r_i : \mathsf{int} \qquad\qquad \psi; \Gamma \vdash v : \mathsf{code}(\Gamma)$$

and that $\Gamma_2 = \Gamma$. That $\psi; \Gamma \vdash v : \mathsf{code}(\Gamma)$ means that $\widehat{H}(\widehat{R}(v)) = J$ for some instruction sequence J for which $\psi \vdash J : \mathsf{code}(\Gamma)$ is derivable, and that $R(i) = n \in \mathbb{Z}$. There are two cases:

  - ⋆ If $n = 0$ then rule COND-1 gives $(H, R, \mathsf{if}\ r_i\ \mathsf{jump}\ v; I') \to (H, R, J)$, and we can derive $\vdash (H, R, J)$ using the S-SEQ rule.
  - ⋆ If $n \neq 0$ then rule COND-2 gives $(H, R, \mathsf{if}\ r_i\ \mathsf{jump}\ v : I') \to (H, R, I')$, and we can derive $\vdash (H, R, I')$ using the S-SEQ rule.

This completes the proof.

References

[1] Pierce, B.C. *Advanced Topics in Types and Programming Languages.* MIT Press, 2004.