Proof of Type Safety for Extended MinML

Amr Sabry
March 26, 2001

1 Definitions

Syntax:

\[ t ::= \text{int} | \text{bool} | \alpha | t \to t | \text{rec } \alpha.t | t \text{ ref} \]

\[ e ::= x | n | o(e_1, \ldots, e_n) | \text{true} | \text{false} | \text{if } e \text{ e e} | \lambda x^t : t.e \ | ee \ | \text{roll } e \ | \text{unroll } e | \ell \ | \text{ref } e \ | \ell.e \ | e := e \]

Typing judgments for expressions are of the form \( \Lambda; \Gamma \vdash e : t \)
Typing judgment for memory is of the form \( \vdash M : \Lambda \).
Evaluation judgments are of the form \( (M, e) \rightsquigarrow (M', e') \)

2 Preservation

Lemma 2.1 (Preservation) If \( \Lambda; \bullet \vdash e : t \) and \( \vdash M : \Lambda \) and \( (M, e) \rightsquigarrow (M', e') \) then there exists a \( \Lambda' \trianglerighteq \Lambda \) such that \( \vdash M' : \Lambda' \) and \( \Lambda'; \bullet \vdash e' : t \).

Proof. The proof is by induction on the evaluation judgments. We present one case only:

- Case \( (M, (+e_1, e_2)) \rightsquigarrow (M', +e'_1, e'_2) \) because \( (M, e_1) \rightsquigarrow (M', e'_1) \). By assumption:
  - \( \Lambda; \bullet \vdash +e_1, e_2 : t \): By inversion \( t \) must be \text{int} and \( \Lambda; \bullet \vdash e_1 : \text{int} \) and \( \Lambda; \bullet \vdash e_2 : \text{int} \).
  - \( \vdash M : \Lambda \)

The evaluation judgment \( (M, e_1) \rightsquigarrow (M', e'_1) \) is shorter, and all the assumptions in the statement of the lemma are satisfied, hence we can apply the inductive hypothesis to conclude that there exists a \( \Lambda' \trianglerighteq \Lambda \) such that \( \vdash M' : \Lambda' \) and \( \Lambda'; \bullet \vdash e'_1 : \text{int} \). To finish this case, we need to conclude that \( \Lambda'; \bullet \vdash +e'_1, e'_2 : \text{int} \), but this is immediate.