Australian National University

COMP3610/6361 Principles of Programming Languages

Peter Höfner

Aug 1, 2023

Section 6

Typing for Call-By-Value

Typing Functions - TypeEnvironment

- so far Γ ranges over TypeEnv, the finite partial function from $\mathbb{L} \rightharpoonup \mathbb{Z}$
- with functions, it summarises assumptions on the types of variables
- type environments Γ are now pairs of a Γ_{loc} ($\mathbb{L} \rightharpoonup \mathbb{Z}$) and a Γ_{var}, a partial function from $\mathbb{X}$ to T ($\mathbb{X} \rightharpoonup T$).

  For example, $\Gamma_{loc} = \{l_1 : \text{intref}\}$ and $\Gamma_{var} = \{x : \text{int}, y : \text{bool} \rightarrow \text{int}\}$.
- $\text{dom}(\Gamma) = \text{dom}(\Gamma_{loc}) \cup \text{dom}(\Gamma_{var})$.
- notation: if $x \notin \text{dom}(\Gamma_{var})$, we write $\Gamma, x : T$, which adds $x : T$ to Γ_{var}

Typing Functions

(var) $\Gamma \vdash x : T$ if $\Gamma(x) = T$

(fn) $$\frac{\Gamma, x : T \ \vdash E : T'}{\Gamma \vdash (\textbf{fn}\ x : T \Rightarrow E) : T \to T'}$$

(app) $$\frac{\Gamma \vdash E_1 : T \to T' \qquad \Gamma \vdash E_2 : T}{\Gamma \vdash E_1\ E_2 : T'}$$

Typing Functions – Example I

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x:\mathsf{int} \vdash x:\mathsf{int}}\,(\text{VAR}) \qquad \dfrac{}{x:\mathsf{int}\vdash 2:\mathsf{int}}\,(\text{INT})}{x:\mathsf{int}\vdash x+2:\mathsf{int}}\,(\text{OP}+)}{\{\}\vdash (\textbf{fn}\ x:\mathsf{int}\Rightarrow x+2):\mathsf{int}\to\mathsf{int}}\,(\text{FN}) \qquad \dfrac{}{\{\}\vdash 2:\mathsf{int}}\,(\text{INT})}{\{\}\vdash (\textbf{fn}\ x:\mathsf{int}\Rightarrow x+2)\ 2:\mathsf{int}}\,(\text{APP})$$

Typing Functions – Example II

Determine the type of

$$(\textbf{fn}\ x : \mathsf{int} \to \mathsf{int} \Rightarrow x\ (\textbf{fn}\ x : \mathsf{int} \Rightarrow x)\ 3)$$

Properties Typing

We only consider *closed* programs, with *no* free variables.

Theorem (Progress)

If E closed, $\Gamma \vdash E : T$ and $\mathrm{dom}(\Gamma) \subseteq \mathrm{dom}(s)$ then either E is a value or there exist E' and s' such that $\langle E\,,\, s\rangle \longrightarrow \langle E'\,,\, s'\rangle$.

There are more configuration that get stuck, e.g. $(3\ 4)$.

Theorem (Type Preservation)

If E *closed,* $\Gamma \vdash E : T$*, dom*$(\Gamma) \subseteq$ *dom*(s) *and* $\langle E\,,\, s\rangle \longrightarrow \langle E'\,,\, s'\rangle$ *then* $\Gamma \vdash E' : T$ *and dom*$(\Gamma) \subseteq$ *dom*(s')*.*

Proving Type Preservation

Theorem (Type Preservation)

If E closed, $\Gamma \vdash E : T$, dom$(\Gamma) \subseteq$ dom(s) and $\langle E\ ,\ s\rangle \longrightarrow \langle E'\ ,\ s'\rangle$ then $\Gamma \vdash E' : T$ and dom$(\Gamma) \subseteq$ dom(s').

Proof outline.

Choose

$$\Phi(E, s, E', s') = \forall \Gamma, T.\ \big(\Gamma \vdash E : T \wedge \mathsf{closed}(E) \wedge \mathsf{dom}(\Gamma) \subseteq \mathsf{dom}(s) \\ \Longrightarrow \Gamma \vdash E' : T \wedge \mathsf{closed}(E') \wedge \mathsf{dom}(\Gamma) \subseteq \mathsf{dom}(s')\big)$$

show $\forall E, s, E', s'.\ \langle E\ ,\ s\rangle \longrightarrow \langle E'\ ,\ s'\rangle \Longrightarrow \Phi(E, s, E', s')$, by rule induction □

Proving Type Preservation – Auxiliary Lemma

Lemma (Substitution)

If E *closed,* $\Gamma \vdash E : T$ *and* $\Gamma, x : T \vdash E' : T'$ *with* $x \notin$ *dom*(Γ) *then* $\Gamma \vdash \{E/x\}\, E' : T'$.

Type Safety

Main result: Well-typed programs do not get stuck.

Theorem (Type Safety)

If $\Gamma \vdash E : T$*, dom*$(\Gamma) \subseteq$ *dom*(s)*, and* $\langle E\ ,\ s\rangle \longrightarrow^* \langle E'\ ,\ s'\rangle$ *then either* E' *is a value with* $\Gamma \vdash E' : T$*, or there exist* E''*,* s'' *such that* $\langle E'\ ,\ s'\rangle \longrightarrow \langle E''\ ,\ s''\rangle$*,* $\Gamma \vdash E'' : T$ *and dom*$(\Gamma) \subseteq$ *dom*(s'')*.g*

Here, $\longrightarrow^*$ means arbitrary many steps in the transition system.

Normalisation

Theorem (Normalisation)

In the sublanguage without while loops or store operations, if $\Gamma \vdash E : T$ *and* E *closed then there does not exist an infinite reduction sequence*

$$\langle E\,,\,\{\}\rangle \longrightarrow \langle E_1\,,\,\{\}\rangle \longrightarrow \langle E_2\,,\,\{\}\rangle \longrightarrow \ldots$$

Proof.

See B. Pierce, Types and Programming Languages, Chapter 12. □