

Reinterpret Types

1 INTRO

Here is a formalization of our type system.

2 CORE LANGUAGE

First, we'll define a small core language with basic integers, booleans, and functions.

$$\begin{array}{lll}
 e & ::= & \mathbb{Z} \mid \mathbb{B} \mid x \mid \text{fun } x \rightarrow e \mid e \ e \quad \text{expressions} \\
 x & ::= & (\text{identifiers}) \quad \text{variables} \\
 v & ::= & \mathbb{Z} \mid \mathbb{B} \mid \text{fun } x \rightarrow e \mid x \quad \text{values} \\
 \tau & ::= & \text{int} \mid \text{bool} \mid \tau \rightarrow \tau \quad \text{types}
 \end{array}$$

Fig. 1. Core language grammar

The typing rules of the system is defined as following:

Definition 2.1 (Typing rules).

- (1) $\models e : \text{int}$ iff $e \Longrightarrow v, v \in \mathbb{Z}$.
- (2) $\models e : \text{bool}$ iff $e \Longrightarrow v, v \in \mathbb{B}$.
- (3) $\models e : \tau_1 \rightarrow \tau_2$ iff $\forall v$ such that $\models v : \tau_1, \models e \ v : \tau_2$.

3 LANGUAGE EXTENSIONS

Next, we will define a couple of languages extensions and their corresponding typing rules.

$$\begin{array}{lll}
 e & ::= & \dots \mid a \quad \text{expressions} \\
 v & ::= & \dots \mid a \quad \text{values} \\
 \tau & ::= & \dots \mid \alpha \mid \tau \cup \tau \mid \tau \cap \tau \mid \{\tau \mid e\} \mid (x : \tau) \rightarrow \tau \mid \mu\alpha. \tau \quad \text{types}
 \end{array}$$

Fig. 2. Extended language grammar

Definition 3.1 (More typing rules).

- (1) $\models e : \alpha_i$ iff $e \Longrightarrow a_i$.
- (2) $\models e : \tau_1 \cup \tau_2$ iff $\models e : \tau_1$ or $\models e : \tau_2$.
- (3) $\models e : \tau_1 \cap \tau_2$ iff $\models e : \tau_1$ and $\models e : \tau_2$.
- (4) $\models e : \{\tau \mid p\}$ iff $\models e : \tau$ and $p \ e \Longrightarrow \text{true}$.
- (5) $\models e : (x : \tau_1) \rightarrow \tau_2$ iff $\forall v$ such that $\models v : \tau_1, \models e \ v : \tau_2[v/x]$.
- (6) $\models e : \mu\alpha. \tau$ iff $e : \tau[\mu\alpha. \tau/\alpha]$.

We will now extend the language with records.

$$\begin{array}{lll}
 e & ::= & \dots \mid \{\overline{\ell = e}\}^{\{\bar{\ell}\}} \quad \text{expressions} \\
 v & ::= & \dots \mid \{\overline{\ell = v}\}^{\{\bar{\ell}\}} \quad \text{values} \\
 \tau & ::= & \dots \mid \{\overline{\ell : \tau}\} \quad \text{types}
 \end{array}$$

Fig. 3. Extended language grammar (with records)

Definition 3.2 (Record typing rules).

- (1) $\models e : \{\ell_1 : \tau_1, \dots, \ell_m : \tau_m\}$ iff $e \Longrightarrow \{\ell_1 = v_1, \dots, \ell_m = v_m, \dots, \ell_n = v_n\}^{\{\ell_1, \dots, \ell_m\}}$ where $\models v_i : \tau_i$ for $i \in \{1, \dots, m\}$, $n \geq m$

4 TYPE AS VALUES

In this section, we will demonstrate how to represent each type using a tuple of two functions, generator and checker.

Definition 4.1 (Semantic interpretation of types). We define the semantic interpretation of types as $\llbracket \tau \rrbracket$, where $\llbracket \tau \rrbracket = \langle \text{generator}(\tau), \text{checker}(\tau) \rangle$.

Definition 4.2 (Defining Generator in the core language).

- (1) $\text{generator}(\text{int}) : \text{pick } n \in \mathbb{Z}$.
- (2) $\text{generator}(\text{bool}) : \text{pick } b \in \mathbb{B}$.
- (3) $\text{generator}(\tau_1 \rightarrow \tau_2) : \text{pick } \lambda x.e$ such that $\forall v$ where $\models v : \tau_1$, $\models (\lambda x.e) v : \tau_2$.

Definition 4.3 (Defining Checker in the core language).

- (1) $\text{checker}(\text{int}) : e \Longrightarrow v$ and $v \in \mathbb{Z}$.
- (2) $\text{checker}(\text{bool}) : e \Longrightarrow v$ and $v \in \mathbb{B}$.
- (3) $\text{checker}(\tau_1 \rightarrow \tau_2) : \forall v$ if $\models v : \tau_1$, then $\models (\lambda x.e) v : \tau_2$.