# Type Checking

Barış Aktemur
CS321 Programming Languages
Ozyegin University

Many programming languages have the notion of *static typing*. With this feature, the input program is passed through a type-checking phase *before* the code is run. The idea is to catch potential errors early, hopefully before the program is executed and not *during* the runtime. This is valuable especially when the problem resides in a hard-to-reach point in the program. You would prefer to detect and fix an error before you deliver your software, rather than get a crash report from the client!

In this lecture, we will see how we can add a type-checking phase to Deve. We will build our code on top of Deve 3.0, available at `https://github.com/aktemur/cs321/tree/master/Deve-3.0`.

Before going any further, you should read PLC sections 4.1–4.9, and 5.4. There are some syntactic differences between the book and our notes/code, but these are small.

## Type Rules

In our Deve language, an expression may have one of the following 4 types:

- An integer
- A boolean
- A pair (of two other types)
- A function (from a type to another type)

Here is the formal (mathematical) definition of types:

$$\rho \in \text{TypeEnv} = \text{Name} \mapsto \text{Type}$$
$$\tau \in \text{Type} := \text{int} \mid \text{bool} \mid (\tau_1 \to \tau_2) \mid (\tau_1 \times \tau_2)$$

We require the programmer to annotate functions with type information. So the syntax of functions is extended to include the input and output types. The full syntax of the language is given in Figure 1. Logical rules for type-checking are shown in Figure 2.

> The type-checked version of the language is Deve 4.0. It is available at `https://github.com/aktemur/cs321/tree/master/Deve-4.0`.

$$
\begin{aligned}
i &\in \text{Int} \\
b &\in \text{Bool} ::= \texttt{true} \mid \texttt{false} \\
x &\in \text{Name} \\
e &\in \text{Exp} ::= i \mid b \mid x \\
&\quad \mid e_1 \texttt{ + } e_2 \mid e_1 \texttt{ - } e_2 \mid e_1 \texttt{ * } e_2 \mid e_1 \texttt{ / } e_2 \\
&\quad \mid e_1 \texttt{ < } e_2 \mid e_1 \texttt{ <= } e_2 \mid \texttt{(}e_1\texttt{,}e_2\texttt{)} \\
&\quad \mid \texttt{if } e_1 \texttt{ then } e_2 \texttt{ else } e_3 \\
&\quad \mid \texttt{let } x \texttt{ = } e_1 \texttt{ in } e_2 \\
&\quad \mid \texttt{fun (}x\texttt{:}\tau\texttt{) -> } e \\
&\quad \mid e_1\, e_2 \\
&\quad \mid \texttt{match } e_1 \texttt{ with (}x\texttt{,}y\texttt{) -> } e_2 \\
&\quad \mid \texttt{let rec f (}x\texttt{:}\tau_1\texttt{) :}\tau_2 \texttt{ = } e_1 \texttt{ in } e_2
\end{aligned}
$$

Figure 1: The abstract syntax of the Deve language.

$$\frac{}{\rho \vdash i : \text{int}} \quad \text{(rule 1)} \qquad \frac{}{\rho \vdash b : \text{bool}} \quad \text{(rule 2)} \qquad \frac{\rho(x) = \tau}{\rho \vdash x : \tau} \quad \text{(rule 3)}$$

$$\frac{\rho \vdash e_1 : \text{int} \qquad \rho \vdash e_2 : \text{int}}{\rho \vdash e_1 \texttt{ + } e_2 : \text{int}} \quad \text{(rule 4)} \qquad \text{(and similarly for } \texttt{-}, \texttt{*}, \texttt{/})$$

$$\frac{\rho \vdash e_1 : \text{int} \qquad \rho \vdash e_2 : \text{int}}{\rho \vdash e_1 \texttt{ < } e_2 : \text{bool}} \quad \text{(rule 5)} \qquad \text{(and similarly for } \texttt{<=})$$

$$\frac{\rho \vdash e_1 : \tau_1 \qquad \rho \vdash e_2 : \tau_2}{\rho \vdash \texttt{(}e_1\texttt{,}e_2\texttt{)} : (\tau_1 \times \tau_2)} \quad \text{(rule 6)}$$

$$\frac{\rho \vdash e_1 : \text{bool} \qquad \rho \vdash e_2 : \tau \qquad \rho \vdash e_3 : \tau}{\rho \vdash \texttt{if } e_1 \texttt{ then } e_2 \texttt{ else } e_3 : \tau} \quad \text{(rule 7)}$$

$$\frac{\rho \vdash e_1 : \tau_1 \qquad [x \mapsto \tau_1] + \rho \vdash e_2 : \tau_2}{\rho \vdash \texttt{let } x \texttt{ = } e_1 \texttt{ in } e_2 : \tau_2} \quad \text{(rule 8)}$$

$$\frac{[x \mapsto \tau_1] + \rho \vdash e : \tau_2}{\rho \vdash \texttt{fun (}x\texttt{:}\tau_1\texttt{) -> } e : (\tau_1 \to \tau_2)} \quad \text{(rule 9)}$$

$$\frac{\rho \vdash e_1 : (\tau_2 \to \tau_1) \qquad \rho \vdash e_2 : \tau_2}{\rho \vdash e_1\, e_2 : \tau_1} \quad \text{(rule 10)}$$

$$\frac{\rho \vdash e_1 : (\tau_1 \times \tau_2) \qquad [x \mapsto \tau_1, y \mapsto \tau_2] + \rho \vdash e_2 : \tau}{\rho \vdash \texttt{match } e_1 \texttt{ with (}x\texttt{,}y\texttt{) -> } e_2 : \tau} \quad \text{(rule 11)}$$

$$\frac{[f \mapsto (\tau_1 \to \tau_2), x \mapsto \tau_1] + \rho \vdash e_1 : \tau_2 \qquad [f \mapsto (\tau_1 \to \tau_2)] + \rho \vdash e_2 : \tau}{\rho \vdash \texttt{let rec f (}x\texttt{:}\tau_1\texttt{) :}\tau_2 \texttt{ = } e_1 \texttt{ in } e_2 : \tau} \quad \text{(rule 12)}$$

Figure 2: Typing rules for the Deve language.