

CS738: Advanced Compiler Optimizations

Typed Arithmetic Expressions

Amey Karkare

karkare@cse.iitk.ac.in

<http://www.cse.iitk.ac.in/~karkare/cs738>

Department of CSE, IIT Kanpur



Reference Book

Types and Programming Languages by Benjamin C. Pierce

Recap: Untyped Arithmetic Expression Language

$t :=$	– <i>terms</i>
true	– <i>constant true</i>
false	– <i>constant false</i>
if t then t else t	– <i>conditional</i>
0	– <i>constant zero</i>
succ t	– <i>successor</i>
pred t	– <i>predecessor</i>
iszero t	– <i>zero test</i>

Recap: The Set of Values

$v :=$	– <i>values</i>
true	– <i>value true</i>
false	– <i>value false</i>
0	– <i>value zero</i>
succ v	– <i>successor value</i>

Let's add Types to the Language

$T :=$

Bool	– <i>Types</i>
Nat	– <i>Booleans</i>
	– <i>Natural Numbers</i>

The Typing Relation

- ▶ A set of rules assigning types to terms
- ▶ $\vdash t : T$ denotes “term t has type T ”

$$\begin{array}{c} 0 : \text{Nat} \\[1em] \frac{t_1 : \text{Nat}}{\text{succ } t_1 : \text{Nat}} \\[1em] \frac{t_1 : \text{Nat}}{\text{pred } t_1 : \text{Nat}} \\[1em] \frac{t_1 : \text{Nat}}{\text{iszero } t_1 : \text{Bool}} \end{array}$$

The Typing Relation (contd...)

- ▶ A set of rules assigning types to terms
- ▶ $\vdash t : T$ denotes “term t has type T ”

$$\begin{array}{c} \text{true} : \text{Bool} \\[1em] \text{false} : \text{Bool} \\[1em] \frac{t_1 : \text{Bool} \quad t_2 : T \quad t_3 : T}{\text{if } t_1 \text{ then } t_2 \text{ else } t_3 : T} \end{array}$$

The Typing Relation: Definition

- ▶ The *typing relation* for arithmetic expressions is the smallest binary relation between terms and types satisfying all instances of the rules defined earlier.
- ▶ A term t is *typeable* (or *well typed*) if there is some T such that $t : T$.

Inversion of the Typing Relation

- ▶ If $\vdash 0 : R$, then $R = \text{Nat}$.
- ▶ If $\vdash \text{succ } t_1 : R$, then $R = \text{Nat}$ and $\vdash t_1 : \text{Nat}$.
- ▶ If $\vdash \text{pred } t_1 : R$, then $R = \text{Nat}$ and $\vdash t_1 : \text{Nat}$.
- ▶ If $\vdash \text{iszero } t_1 : R$, then $R = \text{Bool}$ and $\vdash t_1 : \text{Nat}$.
- ▶ If $\vdash \text{true} : R$, then $R = \text{Bool}$.
- ▶ If $\vdash \text{false} : R$, then $R = \text{Bool}$.
- ▶ If $\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 : R$, then
 - ▶ $\Gamma \vdash t_1 : \text{Bool}$
 - ▶ $\Gamma \vdash t_2 : R$
 - ▶ $\Gamma \vdash t_3 : R$

Uniqueness of Types

- ▶ Every term t has at most one type.
- ▶ If t is typeable, then its type is unique.
- ▶ Moreover, there is just one derivation of this typing built from the inference rules.

Safety = Preservation + Progress

- ▶ The type system is *safe* (also called *sound*)
- ▶ Well-typed programs do not “go wrong.”
 - ▶ Do not reach a “stuck state.”
- ▶ **Progress:** A well-typed term is not stuck.
 - ▶ If $\vdash t : T$, then t is either a value or there exists some t' such that $t \rightarrow t'$.
- ▶ **Preservation:** If a well-typed term takes a step of evaluation, then the resulting term is also well-typed.
 - ▶ If $\vdash t : T$ and $t \rightarrow t'$, then $\vdash t' : T$.