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 :=	– *terms*
`true`	– *constant true*
`false`	– *constant false*
`if` t `then` t `else` t	– *conditional*
0	– *constant zero*
`succ` t	– *successor*
`pred` t	– *predecessor*
`iszero` t	– *zero test*

Recap: The Set of Values

v :=		– *values*
	`true`	– *value true*
	`false`	– *value false*
	0	– *value zero*
	`succ` v	– *successor value*

Let's add Types to the Language

$T :=$ – *Types*

Let's add Types to the Language

T :=	– *Types*
Bool	– *Booleans*

Let's add Types to the Language

T :=		– *Types*
	Bool	– *Booleans*
	Nat	– *Natural Numbers*

The Typing Relation

- A set of rules assigning types to terms

The Typing Relation

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

The Typing Relation

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

The Typing Relation

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

$$0 : \mathrm{Nat}$$

The Typing Relation

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

$$0 : \mathrm{Nat}$$

$$\frac{t_1 : \mathrm{Nat}}{\texttt{succ}\ t_1 : \mathrm{Nat}}$$

The Typing Relation

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

$$0 : \mathsf{Nat}$$

$$\frac{t_1 : \mathsf{Nat}}{\mathtt{succ}\ t_1 : \mathsf{Nat}}$$

$$\frac{t_1 : \mathsf{Nat}}{\mathtt{pred}\ t_1 : \mathsf{Nat}}$$

The Typing Relation

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

$$0 : \mathsf{Nat}$$

$$\frac{t_1 : \mathsf{Nat}}{\mathtt{succ}\ t_1 : \mathsf{Nat}}$$

$$\frac{t_1 : \mathsf{Nat}}{\mathtt{pred}\ t_1 : \mathsf{Nat}}$$

$$\frac{t_1 : \mathsf{Nat}}{\mathtt{iszero}\ t_1 : \mathsf{Bool}}$$

The Typing Relation (contd...)

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

$$\texttt{true} : \text{Bool}$$

The Typing Relation (contd...)

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

$$\texttt{true} : \mathsf{Bool}$$

$$\texttt{false} : \mathsf{Bool}$$

The Typing Relation (contd...)

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

$$\texttt{true} : \mathrm{Bool}$$

$$\texttt{false} : \mathrm{Bool}$$

$$\frac{t_1 : \mathrm{Bool} \qquad t_2 : T \qquad t_3 : T}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 : T}$$

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.

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 $R = \text{Nat}$.

Inversion of the Typing Relation

- If $\vdash 0 : R$, then $R = \text{Nat}$.
- If $\vdash \texttt{succ}\ t_1 : R$, then $R = \text{Nat}$ and $\vdash t_1 : \text{Nat}$.

Inversion of the Typing Relation

- If $\vdash 0 : R$, then $R = \text{Nat}$.
- If $\vdash \texttt{succ}\ t_1 : R$, then $R = \text{Nat}$ and $\vdash t_1 : \text{Nat}$.
- If $\vdash \texttt{pred}\ t_1 : R$, then $R = \text{Nat}$ and $\vdash t_1 : \text{Nat}$.

Inversion of the Typing Relation

- If $\vdash 0 : R$, then $R = \text{Nat}$.
- If $\vdash \texttt{succ}\ t_1 : R$, then $R = \text{Nat}$ and $\vdash t_1 : \text{Nat}$.
- If $\vdash \texttt{pred}\ t_1 : R$, then $R = \text{Nat}$ and $\vdash t_1 : \text{Nat}$.
- If $\vdash \texttt{iszero}\ t_1 : R$, then $R = \text{Bool}$ and $\vdash t_1 : \text{Nat}$.

Inversion of the Typing Relation

- If $\vdash 0 : R$, then $R = \text{Nat}$.
- If $\vdash \texttt{succ}\ t_1 : R$, then $R = \text{Nat}$ and $\vdash t_1 : \text{Nat}$.
- If $\vdash \texttt{pred}\ t_1 : R$, then $R = \text{Nat}$ and $\vdash t_1 : \text{Nat}$.
- If $\vdash \texttt{iszero}\ t_1 : R$, then $R = \text{Bool}$ and $\vdash t_1 : \text{Nat}$.
- If $\vdash \texttt{true} : R$, then $R = \text{Bool}$.

Inversion of the Typing Relation

- If $\vdash 0 : R$, then $R = \text{Nat}$.
- If $\vdash \texttt{succ}\ t_1 : R$, then $R = \text{Nat}$ and $\vdash t_1 : \text{Nat}$.
- If $\vdash \texttt{pred}\ t_1 : R$, then $R = \text{Nat}$ and $\vdash t_1 : \text{Nat}$.
- If $\vdash \texttt{iszero}\ t_1 : R$, then $R = \text{Bool}$ and $\vdash t_1 : \text{Nat}$.
- If $\vdash \texttt{true} : R$, then $R = \text{Bool}$.
- If $\vdash \texttt{false} : R$, then $R = \text{Bool}$.

Inversion of the Typing Relation

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

Inversion of the Typing Relation

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

Inversion of the Typing Relation

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

Inversion of the Typing Relation

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

Uniqueness of Types

- Every term t has at most one type.
- If t is typeable, then its type is unique.

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*)

Safety = Preservation + Progress

- The type system is *safe* (also called *sound*)
- Well-typed programs do not “go wrong.”

Safety = Preservation + Progress

- The type system is *safe* (also called *sound*)
- Well-typed programs do not "go wrong."
  - Do not reach a "stuck state."

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.

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 $t \rightarrow t'$.

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.

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 $\vdash t' : T$.