

CS738: Advanced Compiler Optimizations

Types and Program Analysis

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

Type: Definition

type

/tʌɪp/

noun

1. a category of people or things having common characteristics.
"this type of heather grows better in a drier habitat"
synonyms: kind, sort, variety, class, category, classification, group, set, bracket, genre, genus, species, family, order, breed, race, strain; More
2. a person or thing exemplifying the ideal or defining characteristics of something.
"she characterized his witty sayings as the type of modern wisdom"
synonyms: epitome, quintessence, essence, perfect example, archetype, model, pattern, paradigm, exemplar, embodiment, personification, avatar; prototype
"she characterized his witty sayings as the type of modern wisdom"

Types in Programming

- ▶ A collection of *values*



- ▶ The operations that are permitted on these values

Type System

- ▶ A collection of rules for checking the correctness of usages of types
 - ▶ “Consistency” of programs

The World of Programming Languages

- ▶ Typed
 - ▶ C, C++, Java, Python, ...
- ▶ Untyped
 - ▶ Assembly, *any other?*

The World of Programming Languages

	Statically Typed	Dynamically Typed
Strongly Typed	ML, Haskell, Pascal (almost), Java (almost)	Lisp, Scheme
Weakly Typed	C, C++	Perl

Applications of Type-based Analyses

- ▶ Error Detection
 - ▶ Language Safety
 - ▶ Verification
- ▶ Abstraction
- ▶ Documentation
- ▶ Maintenance
- ▶ Efficiency

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>

Syntax: Inductive Definition

The set of *terms* is the smallest set \mathcal{T} such that

1. $\{\text{true}, \text{false}, 0\} \subseteq \mathcal{T}$
2. if $t_1 \in \mathcal{T}$, then $\{\text{succ } t_1, \text{pred } t_1, \text{iszero } t_1\} \subseteq \mathcal{T}$
3. if $t_1 \in \mathcal{T}$, $t_2 \in \mathcal{T}$, and $t_3 \in \mathcal{T}$ then if t_1 then t_2 else $t_3 \in \mathcal{T}$

Syntax: Inference Rules

The set of *terms*, \mathcal{T} is defined by the following rules:

$$\begin{array}{c} \text{true} \in \mathcal{T} \qquad \text{false} \in \mathcal{T} \qquad 0 \in \mathcal{T} \\[10pt] \frac{t_1 \in \mathcal{T}}{\text{succ } t_1 \in \mathcal{T}} \qquad \frac{t_1 \in \mathcal{T}}{\text{pred } t_1 \in \mathcal{T}} \qquad \frac{t_1 \in \mathcal{T}}{\text{iszero } t_1 \in \mathcal{T}} \\[10pt] \frac{t_1 \in \mathcal{T} \quad t_2 \in \mathcal{T} \quad t_3 \in \mathcal{T}}{\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \in \mathcal{T}} \end{array}$$

Concrete Syntax

$$\begin{aligned} \mathcal{S}_0 &= \emptyset \\ \mathcal{S}_{i+1} &= \{\text{true}, \text{false}, 0\} \\ &\quad \cup \{\text{succ } t_1, \text{pred } t_1, \text{iszero } t_1 \mid t_1 \in \mathcal{S}_i\} \\ &\quad \cup \{\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \mid t_1, t_2, t_3 \in \mathcal{S}_i\} \end{aligned}$$

Let $\mathcal{S} = \bigcup_i \mathcal{S}_i$.
Then, $\mathcal{T} = \mathcal{S}$.

Induction on Terms

- ▶ Any $t \in \mathcal{T}$
 - ▶ Either a ground term, i.e. $\in \{\text{true}, \text{false}, 0\}$
 - ▶ Or is created from some smaller terms $\in \mathcal{T}$
- ▶ Allows for inductive definitions and inductive proofs.
- ▶ Three sample inductive properties
 - ▶ $\text{Consts}(t)$
 - ▶ $\text{size}(t)$
 - ▶ $\text{depth}(t)$

Consts

- ▶ The set of constants in a term t .

$$\begin{aligned}\text{Consts}(\text{true}) &= \{\text{true}\} \\ \text{Consts}(\text{false}) &= \{\text{false}\} \\ \text{Consts}(0) &= \{0\} \\ \text{Consts}(\text{succ } t) &= \text{Consts}(t) \\ \text{Consts}(\text{pred } t) &= \text{Consts}(t) \\ \text{Consts}(\text{iszero } t) &= \text{Consts}(t) \\ \text{Consts}(\text{if } t_1 \text{ then } t_2 \text{ else } t_3) &= \text{Consts}(t_1) \\ &\quad \cup \text{Consts}(t_2) \\ &\quad \cup \text{Consts}(t_3)\end{aligned}$$

size

- ▶ The number of nodes in the abstract syntax tree of a term t .

$$\begin{aligned}\text{size}(\text{true}) &= 1 \\ \text{size}(\text{false}) &= 1 \\ \text{size}(0) &= 1 \\ \text{size}(\text{succ } t) &= \text{size}(t) + 1 \\ \text{size}(\text{pred } t) &= \text{size}(t) + 1 \\ \text{size}(\text{iszero } t) &= \text{size}(t) + 1 \\ \text{size}(\text{if } t_1 \text{ then } t_2 \text{ else } t_3) &= \text{size}(t_1) + \text{size}(t_2) + \text{size}(t_3)\end{aligned}$$

depth

- ▶ The maximum depth of the abstract syntax tree of a term t .
- ▶ Equivalently, the smallest i such that $t \in \mathcal{S}_i$.

$$\begin{aligned}\text{depth}(\text{true}) &= 1 \\ \text{depth}(\text{false}) &= 1 \\ \text{depth}(0) &= 1 \\ \text{depth}(\text{succ } t) &= \text{depth}(t) + 1 \\ \text{depth}(\text{pred } t) &= \text{depth}(t) + 1 \\ \text{depth}(\text{iszero } t) &= \text{depth}(t) + 1 \\ \text{depth}(\text{if } t_1 \text{ then } t_2 \text{ else } t_3) &= \max(\text{depth}(t_1) + \text{depth}(t_2) \\ &\quad + \text{depth}(t_3)) + 1\end{aligned}$$

A Simple Property of Terms

- ▶ The number of distinct constants in a term t is no greater than the size of t .

$$|\text{Consts}(t)| \leq \text{size}(t)$$

- ▶ **Proof:** Exercise.

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>

Small-step Operational Semantics

- ▶ $t \rightarrow t'$ denotes “ t evaluates to t' in one step”

if true then t_2 else $t_3 \rightarrow t_2$

if false then t_2 else $t_3 \rightarrow t_3$

$$\frac{t_1 \rightarrow t'_1}{\text{if } t_1 \text{ then } t_2 \text{ else } t_3 \rightarrow \text{if } t'_1 \text{ then } t_2 \text{ else } t_3}$$

Small-step Operational Semantics (contd. . .)

- ▶ $t \rightarrow t'$ denotes “ t evaluates to t' in one step”

$$\frac{t_1 \rightarrow t'_1}{\text{succ } t_1 \rightarrow \text{succ } t'_1}$$

pred 0 \rightarrow 0

pred (succ v) $\rightarrow v$

$$\frac{t_1 \rightarrow t'_1}{\text{pred } t_1 \rightarrow \text{pred } t'_1}$$

Small-step Operational Semantics (contd. . .)

- ▶ $t \rightarrow t'$ denotes “ t evaluates to t' in one step”

$\text{iszero } 0 \rightarrow \text{true}$

$\text{iszero } (\text{succ } v) \rightarrow \text{false}$

$$\frac{t_1 \rightarrow t'_1}{\text{iszero } t_1 \rightarrow \text{iszero } t'_1}$$

Normal Form

- ▶ A term is t in normal form if no evaluation rule applies to it.
- ▶ In other words, there is no t' such that $t \rightarrow t'$.

Evaluation Sequence

- ▶ An evaluation sequence starting from a term t is a (finite or infinite) sequence of terms t_1, t_2, \dots , such that

$t \rightarrow t_1$

$t_1 \rightarrow t_2$

etc.

Stuck Term

- ▶ A term is said to be **stuck** if it is a normal form but not a value.
- ▶ A simple notion of “run-time type error”