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
Weekly Typed	C, C++	Perl

Applications of Type-based Analyses

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

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*

Syntax: Inductive Definition

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

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

Syntax: Inference Rules

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

$$\texttt{true} \in \mathcal{T} \qquad \texttt{false} \in \mathcal{T} \qquad 0 \in \mathcal{T}$$

$$\frac{t_1 \in \mathcal{T}}{\texttt{succ}\ t_1 \in \mathcal{T}} \qquad \frac{t_1 \in \mathcal{T}}{\texttt{pred}\ t_1 \in \mathcal{T}} \qquad \frac{t_1 \in \mathcal{T}}{\texttt{iszero}\ t_1 \in \mathcal{T}}$$

$$\frac{t_1 \in \mathcal{T} \qquad t_2 \in \mathcal{T} \qquad t_3 \in \mathcal{T}}{\texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3 \in \mathcal{T}}$$

Concrete Syntax

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

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. $\in \{\mathtt{true}, \mathtt{false}, 0\}$
  - Or is created from some smaller terms $\in \mathcal{T}$
- Allows for inductive definitions and inductive proofs.
- Three sample inductive properties
  - *Consts*(t)
  - *size*(t)
  - *depth*(t)

Consts

- The set of constants in a term t.

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

size

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

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

depth

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

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

A Simple Property of Terms

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

$$|\mathit{Consts}(\mathrm{t})| \leq \mathit{size}(\mathrm{t})$$

- **Proof:** Exercise.

The Set of Values

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

Small-step Operational Semantics

- $\mathrm{t} \to \mathrm{t}'$ denotes "t evaluates to t' in one step"

$$\texttt{if true then } t_2 \texttt{ else } t_3 \to t_2$$

$$\texttt{if false then } t_2 \texttt{ else } t_3 \to t_3$$

$$\frac{t_1 \to t_1'}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 \to \texttt{if } t_1' \texttt{ then } t_2 \texttt{ else } t_3}$$

Small-step Operational Semantics (contd. . .)

- $\mathrm{t} \to \mathrm{t}'$ denotes "t evaluates to t' in one step"

$$\frac{t_1 \to t_1'}{\texttt{succ } t_1 \to \texttt{succ } t_1'}$$

$$\texttt{pred } 0 \to 0$$

$$\texttt{pred } (\texttt{succ } v) \to v$$

$$\frac{t_1 \to t_1'}{\texttt{pred } t_1 \to \texttt{pred } t_1'}$$

Small-step Operational Semantics (contd. . .)

- $\mathrm{t} \rightarrow \mathrm{t}'$ denotes "t evaluates to t' in one step"

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

$$\texttt{iszero}\ (\texttt{succ}\ \mathrm{v}) \rightarrow \texttt{false}$$

$$\frac{\mathrm{t}_1 \rightarrow \mathrm{t}_1'}{\texttt{iszero}\ \mathrm{t}_1 \rightarrow \texttt{iszero}\ \mathrm{t}_1'}$$

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 $\mathrm{t} \rightarrow \mathrm{t}'$.

Evaluation Sequence

- An evaluation sequence starting from a term t is a (finite or infinite) sequence of terms $\mathrm{t}_1, \mathrm{t}_2, \ldots$, such that

$$\mathrm{t} \rightarrow \mathrm{t}_1$$

$$\mathrm{t}_1 \rightarrow \mathrm{t}_2$$

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"