CS3005D Compiler Design

Winter 2024

Lecture #24

Type Checking, Type Conversions

Saleena N
CSED NIT Calicut

February 2024

Type Checking

- Data Type: a set of values plus a set of operations on those values. e.g. set of integers with integer arithmetic operations - addition, subtraction etc.
- Type system of a language consists of a collection of logical rules to assign types to various constructs. Type system is part of the language specification.
- Typechecker: an implementation of the type system. Typechecker assigns a *type expression* to each component of the source program and checks if these type expressions conform to the type system for the source language.

Type System for a sample language

Production	Semantic Rules
$E \rightarrow num$	$E.type = int$
$E \rightarrow id$	$E.type = getType(id.entry)$
$E \rightarrow E_1 + E_2$	$if\ (E_1.type = int\ and\ E_2.type = int)$ $then\ E.type = int\ else\ E.type = type\_error$
$E \rightarrow E_1[E_2]$	$if\ (E_2.type = int\ and\ E_1.type = array(s,\ t))$ $then\ E.type = t\ else\ E.type = type\_error$
$E \rightarrow E_1(E_2)$	$if\ (E_2.type = s\ and\ E_1.type = s \rightarrow t)\ then$ $E.type = t\ else\ E.type = type\_error$
...	...

Type System for a sample language

Production	Semantic Rules
$S \rightarrow id = E$	$if\ (E.type = getType(id.entry))$ $then\ S.type = void\ else\ S.type = type\_error$
$S \rightarrow if(E)\ then\ S_1$	$if\ (E.type = bool)\ then\ S.type = S_1.type$ $else\ S.type = type\_error$
$S \rightarrow while(E)\ S_1$	$if\ (E.type = bool)\ then\ S.type = S_1.type$ $else\ S.type = type\_error$

Note: The type system is not complete. A set of rules for type checking Boolean Expressions are required.

Type Conversions

Conversion from one type to another

- *Implicit*: done automatically by the compiler(also called *coercions*). Type Checker inserts required conversion operations into the intermediate representation. e.g. adding an *int* and a *float* - compiler inserts code to convert the *int* operand to *float* type.
- *Explicit*: conversions written by the programmer(also called *casts*).

Type Checking rules for coercion from int to float

Production	Semantic Rules
$E \rightarrow E_1 + E_2$	*if* $(E_1.type = int$ *and* $E_2.type = int)$ *then* $E.type = int$ *else if* $(E_1.type = int$ *and* $E_2.type = float)$ *then* $E.type = float$ *else if* $(E_1.type = float$ *and* $E_2.type = int)$ *then* $E.type = float$ *else if* $(E_1.type = float$ *and* $E_2.type = float)$ *then* $E.type = float$ *else* $E.type = type\_error$

References

References:

- Aho A.V., Lam M.S., Sethi R., and Ullman J.D. Compilers: Principles, Techniques, and Tools (ALSU). Pearson Education, 2007[1].

Further reading:

- ALSU Chapter 6 - sections 6.5 - 6.5.1, 6.5.2 (only the relevant portions)

[1] some of the topics are from another edition by Aho, Sethi and Ullman