Bahirdar University Faculty of Computing Software Engineering Copiler Design Assignment 2 2018 E.C

Abebaw Abebe --------BDU1504734

Submitted to: M.r Wondimu B.
Subition Dtae: 27-04-2018

Table of Contents Pages

What is a Type Mismatch Error?

1. Introduction

In compiler design, a program must be correct both syntactically and semantically. Syntax analysis checks whether a program follows the grammatical rules of the language, while semantic analysis ensures that the program makes logical and meaningful sense. One of the most important checks performed during semantic analysis is type checking. A type mismatch error occurs when operations in a program involve incompatible data types. These errors do not violate grammar rules, but they violate the meaning and rules defined by the programming language.

2. Definition of Type Mismatch Error

A type mismatch error is a semantic error that occurs when a value of one data type is used in a place where another data type is expected, or when an operation is applied to operands whose data types are not compatible according to the rules of the programming language. Such errors are not detected during syntax analysis but are identified during the semantic analysis phase of the compiler.

3. Why Type Checking is Necessary

Type checking is necessary to prevent meaningless operations and to ensure the correctness of expressions in a program. It helps avoid runtime errors by catching problems early during compilation and improves overall program safety and reliability. Without type checking, a program might compile successfully but fail during execution, which can lead to serious system errors and unpredictable behavior.

4. Detailed Explanation of Type Mismatch Errors

Type mismatch errors can occur in several common programming situations.

4.1 Assignment Type Mismatch

An assignment type mismatch occurs when a value assigned to a variable does not match the variable's declared data type. For example, if an integer variable `count` is assigned the value `12.75`, which is a floating-point number, a type mismatch error occurs. This happens because `count` is declared as an integer, and assigning a floating-point value to it without explicit conversion violates the language's type rules.

4.2 Expression Type Mismatch

An expression type mismatch happens when operators are applied to operands of incompatible data types. For instance, consider an expression where an integer variable is added to a character variable. In strongly typed languages, performing integer addition with a character type may not be allowed. As a result, the compiler reports a type mismatch error because the operand types are not compatible.

4.3 Function Argument Type Mismatch

A function argument type mismatch occurs when the arguments passed to a function do not match the parameter types defined in the function declaration. For example, if a function expects two integer parameters but is called with one integer and one floating-point value, the compiler detects a type mismatch error. This ensures that functions are called correctly according to their defined signatures.

4.4 Return Type Mismatch

A return type mismatch occurs when a function returns a value that does not match its declared return type. For example, if a function is declared to return an integer but returns a floating-point value, the compiler reports a type mismatch error. This check ensures consistency between a function's definition and its actual behavior.

4.5 Relational and Logical Type Mismatch

Relational and logical type mismatches occur when relational or logical operators are used incorrectly. For example, if a conditional statement expects a boolean value but is given an integer expression instead, some programming languages treat this as a type mismatch error. This prevents incorrect logical evaluations and ensures that conditions are meaningful.

5. Detection of Type Mismatch Errors in the Compiler

During the semantic analysis phase, the compiler builds and maintains a symbol table that stores information such as variable names, data types, and scope details. The compiler then checks every assignment statement, expression, and function call to verify that operand types are compatible, assignment types match, and function signatures are respected. If any of these rules are violated, the compiler reports a type mismatch error.

6. Practical Example with Compiler Perspective

Consider a program in which an integer variable `a` and a floating-point variable `b` are declared, and the expression `a = b + 2.5` is evaluated. During semantic analysis, the compiler determines that `b` is a float and `2.5` is also a float, so the result of the expression is a float. Since this floating-point result is being assigned to an integer variable `a`, the compiler detects a type mismatch error.

That is:-

```
int a;
float b;
a = b + 2.5;
```

Compiler's Semantic Analysis:

- ✓ b is float
- ✓ 2.5 is float
- ✓ Expression result is float
- ✓ Assigned to a (int)

Result: **Type mismatch error**

7. Real-World Applications

Type mismatch checking plays a crucial role in real-world software development. In compiler safety, it prevents incorrect operations before execution and saves significant debugging time. In large-scale software systems, such as banking and medical applications, type checking ensures data integrity and prevents serious logical failures. In mobile and embedded systems, it helps avoid crashes caused by incorrect memory usage and improves performance and reliability. Modern programming

languages like Java, Kotlin, and Swift rely heavily on strong type checking to help developers catch errors early in the development process.

8. Role of Type Mismatch Errors in the Compiler Design Process

Type mismatch checking is an essential part of the compiler pipeline. After lexical analysis generates tokens and syntax analysis validates grammar rules, semantic analysis performs type checking, detects type mismatch errors, and checks scope and declarations. Only programs that pass these checks proceed to intermediate code generation, optimization, and target code generation. This ensures that only meaningful and type-safe programs reach the later stages of compilation.

9. Difference Between Syntax Error and Type Mismatch Error

A syntax error occurs during the syntax analysis phase and represents a violation of grammatical rules, such as a missing semicolon, which results in an incorrect program structure. In contrast, a type mismatch error occurs during the semantic analysis phase and represents a violation of meaning, such as assigning a floating-point value to an integer variable. In this case, the program structure is correct, but its meaning is not.

Aspect	Syntax Error	Type Mismatch Error
Phase	Syntax Analysis	Semantic Analysis
Meaning	Grammar violation	Meaning violation
Example	Missing semicolon	Assigning float to int
Program structure	Incorrect	Correct

Conclusion

A type mismatch error is a semantic error caused by using incompatible data types in a program. It is detected during the semantic analysis phase of the compiler and plays a critical role in ensuring program correctness, reliability, and safety. By enforcing strict type rules, compilers prevent meaningless operations and help developers produce high-quality, error-free software.