16. Type Checking for Overloaded Operators

Introduction

Operator overloading allows the same operator symbol (such as +, -, *, ==) to perform different operations depending on the types of its operands.
During **semantic analysis**, the compiler must determine **which version of an operator to apply** and must **detect errors** such as ambiguous usage or incompatible operand types.

Type checking for overloaded operators ensures that:

- The operator is defined for the given operand types
- The correct operator version is selected
- Invalid or ambiguous expressions are reported as semantic errors

Why Operator Overloading Needs Type Checking

Without proper type checking:

- The compiler may choose the wrong operation
- Programs may behave incorrectly
- Errors may appear only at runtime

Semantic analysis prevents this by resolving operator meanings **before code generation**.

Rules for Type Checking Overloaded Operators

1. Operator Definition Rules

Each overloaded operator must have:

- Operator symbol (e.g., +)
- Operand types
- Result type

Example operator table entry:

Operator	Operand Types	Result Type
+	(int, int)	int
+	(float, float)	float
+	(string, string)	string

2. Operand Type Matching

When an operator is used:

- The compiler determines the types of operands
- Searches the operator table for a matching signature

Example:

int a, b;

a + b

→ Matches (int, int) → valid

3. Unique Operator Resolution

- Exactly **one** operator definition must match
- If more than one match exists → **ambiguous operator error**

Example (ambiguous):

operator +(int, float)

operator +(float, int)

x + y   // x:int, y:float

→ Ambiguity error

4. Type Conversion Rules (if allowed)

Some languages allow **implicit conversions** (e.g., int → float).
Resolution priority:

1. Exact match

2. Widening conversions
3. User-defined overloads

If multiple conversions are equally valid → error.

5. Illegal Operator Use

If no operator definition matches operand types:

- Report a **type mismatch error**

Example:

int x;

string y;

x + y   // invalid

Semantic Error Detection

The type checker must detect:

1. Undefined Operator

bool a, b;

a * b   // '*' not defined for bool

2. Ambiguous Operator Use

Multiple operator versions match equally well.

3. Operand Type Mismatch

Operands do not match any operator signature.

Attribute Grammar Representation

Using attribute grammars:

Attributes

- type → data type of expression
- op → operator symbol

Semantic Rule Example