6.110 Computer Language Engineering

Recitation 6: CFG construction

Mar 5, 2025

Weekly updates ←

Introduction to CFG construction

Weekly updates

Phase 2 is due **Friday, Mar 7 (aka TOMORROW)**

Quiz 1 will be held next **Friday, Mar 14**

- **11 AM, 32-123**
- More logistics on Piazza
- Review session **next Wednesday**

Coming up soon...

Mon 3/10	Tue 3/11	Wed 3/12	Thu 3/13	Fri 3/14
No class! 😋		Quiz 1 review	No class	Quiz 1

Weekly updates

Introduction to CFG construction ←

Overview

Many of you asked about CFG and IR

We’ll cover in this recitation how to build IR and CFG from an AST

x86 tutorial merged with tomorrow’s recitation

Parse tree vs AST?

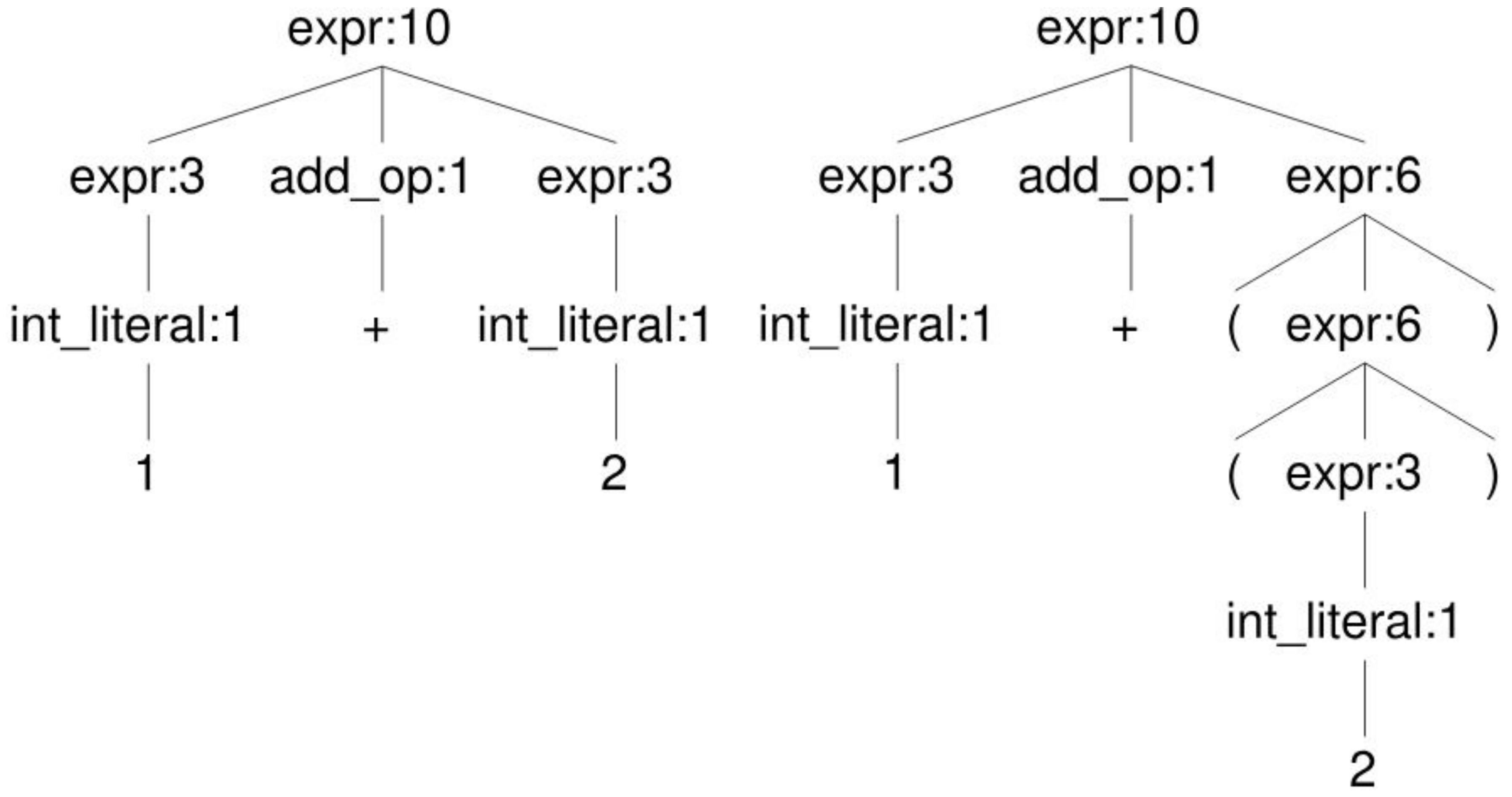
A parse tree may contain irrelevant details

- Parentheses, whitespace and comments

AST is more “**abstract**”

- “`1 + 2`”, “`1+((2))`”, “`1 + /*yo*/ (2)`” have different parse trees but the same AST

Parse tree vs AST?



Parse tree vs AST?

A parse tree may contain irrelevant details

- Parentheses, whitespace and comments

AST is more "**abstract**"

- "`1 + 2`", "`1+((2))`", "`1 + /*yo*/ (2)`"
  have different parse trees but the same AST

You only need a parse tree for Phase 1

You really should have an AST for Phase 2

IR vs AST vs CFG vs SSA… 🤔

IR: umbrella term for any internal representation

A compiler typically has **multiple IRs**

IR vs AST vs CFG vs SSA… 🤔

IR: umbrella term for any internal representation

A compiler typically has **multiple IRs**

- AST + symbol table: good for **semantic analysis**
  - You only need this for Phase 2
- CFG: good for **codegen & optimization**
  - You should have this for Phase 3
  - SSA: a flavor of CFG

IR vs AST vs CFG vs SSA... 🤔

IR: umbrella term for any internal representation

A compiler typically has **multiple IRs**

- AST + symbol table: good for **semantic analysis**
- CFG: good for **codegen & optimization**

Do I need symbol table for CFG?

- CFGs are **local to each method**
- Global symbol table to track global variables and methods

Three ingredients for CFG IR

Variables / virtual registers

Instructions

- Takes a fixed number of operands
- Easy* conversion to assembly

Basic blocks

- Sequence of inst. where control flow never diverges
- Jump and branching only allowed at the end

Demo: AST → CFG

A small subset of Python

Focus on three parts

- Expression
- Conditional
- Statements