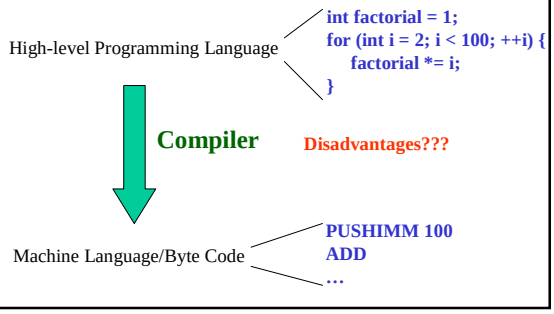
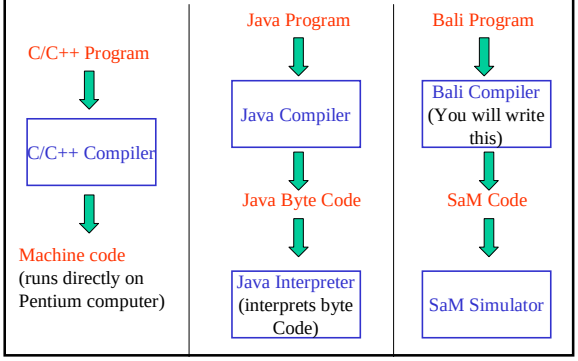
Compilers 101

Why do we need a Compiler?



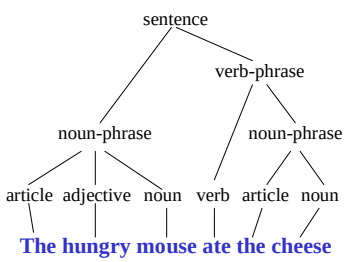
Different Compilers



Languages

- High-level programs are written in languages
  - C++
  - Java
  - Bali
- Compiler needs to understand:
  - Syntax (structure)
  - Semantics (meaning)
- Useful parallel: human languages

Human Language: English



Lexical Analysis
Identify words: easy!

Parsing
Check if words conform to grammar

Semantic Action
Understand/interpret sentence

Simple English Grammar

sentence → noun-phrase verb-phrase

noun-phrase → article [adjective] noun

verb-phrase → verb noun-phrase

- Example of a Context-Free Grammar (CFG)
- Real English grammar much more complex
  - Not Context Free!

Syntax vs. Semantics

sentence → noun-phrase verb-phrase

noun-phrase → article [adjective] noun

verb-phrase → verb noun-phrase

The hungry mouse ate the cheese

The shiny elbow drank the automobile

Syntax = Structure, Semantics = Meaning

Natural vs. Computer Languages

Natural Language	Computer Language
• Lexical Analysis – Break sentence into words	• Lexical Analysis – Break program into tokens (tokenization)
• Parsing – Analyze word arrangement – Discover structure	• Parsing – Analyze token arrangement – Discover structure
• Semantic action – Understand sentence	• Semantic action – Generate target code

Lexical Analysis

- Goal: Divide program into tokens
- Token
  - Smallest element in a language that conveys meaning
- Examples of tokens
  - Operators
  - Names
  - Strings
  - Keywords
  - Numbers

Lexical Analysis (contd.)

- Tokens are typically specified using regular expressions
  - a* = repeat a zero or more times
  - a+ = repeat a one or more times
  - [abc] = choose one of a, b, or c
  - ? = matches any one character
  - ab = a followed by b
- Examples
  - Integer: [0123456789]+ or [0-9]+
  - Variable? (starts with a character and has zero or more following characters/numbers)

Building a Tokenizer

- Can usually identify tokens by "peeking" ahead
- Examples
  - If first character is a letter, then it is a variable
  - If first character is a number, then it is an integer
- Available tools
  - lex (short for lexical analyzer)
  - Java has a class java.io.StreamTokenizer
- For Bali, we will give you the lexical analyzer

Natural vs. Computer Languages

Natural Language	Computer Language
• Lexical Analysis – Break sentence into words	• Lexical Analysis – Break program into tokens (tokenization)
• Parsing – Analyze word arrangement – Discover structure	• Parsing – Analyze token arrangement – Discover structure
• Semantic action – Understand sentence	• Semantic action – Generate target code

Parsing

- Goals
  - Check if input string conforms to grammar (syntax)
  - Build a parse tree or abstract syntax tree for input string that can be used by semantic action (semantics)
- Starting point: grammar

Grammars

- Context Free Grammar (CFG)
  - Anything in single quotes is a token
  - Parenthesis () are used for grouping
  - Brackets [] are used to denote optional
  - | is used to denote choice
- Example: arithmetic expressions
  - Terminals in red, Non-Terminals in blue
  - n and w stand for numeric tokens and variables, resp.

Start symbol → Expr ⟶ Term [('+' | '-') Expr]

Term ⟶ Factor [('*' | '/') Term]

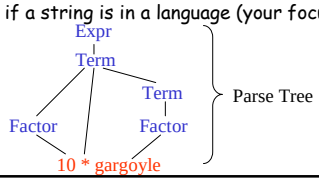
Factor ⟶ n | w | '(' Expr ')'

Grammars (contd.)

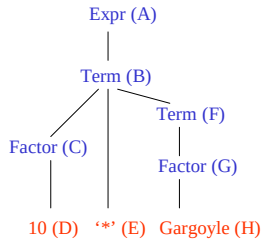
- Grammars define a language
- Two ways in which grammars can be used
  - Generate a string in the language

Expr ⟶ Term
⟶ Factor * Term
⟶ 10 * Term
⟶ 10 * Factor
⟶ 10 * gargoyle

  - Check if a string is in a language (your focus)



Tree Terminology



- A is the root
- A is the parent of B
- B is the parent of C, E, F
- C, E, F are children of B
- B is an ancestor of C, D, E, F, G, H
- C, D, E, F, G, H are descendants of B
- D, E, H are leaves

Ambiguous Grammars

- Consider following grammar
  - ∈ denotes the empty string
  - Aside: what language does grammar produce?

S ⟶ ∈
S ⟶ a S b S
S ⟶ b S a S

- Two ways to derive abab (or equivalently, two valid parse trees for abab)
  - What are they?

Ambiguous Grammars (contd.)

- Ambiguous grammars come up in programming languages too!

```
if (buttonPushed) if (disabled) abortPlan() else launchMissile()
```

```
if (buttonPushed)
  if (disabled)
    abortPlan()
else
  launchMissile()
```

OR

```
if (buttonPushed)
  if (disabled)
    abortPlan()
else
  launchMissile()
```

?

Possible solutions?

Bali grammar will be unambiguous

Natural vs. Computer Languages

Natural Language	Computer Language
• Lexical Analysis – Break sentence into words • Parsing – Analyze word arrangement – Discover structure • Semantic action – Understand sentence	• Lexical Analysis – Break program into tokens (tokenization) • Parsing – Analyze token arrangement – Discover structure • Semantic action – Generate target code