Compiler Construction
CS 541

Mr
Mr
Prof

Raphael
Rafi
See

Finkel
Goldstein

www.cs.uky.edu/~raphael/
courses/CS541.html

Mute Lab:

ssh pen.cs.uky.edu
corr
fail
Compiler outputs

- pure machine code
- augmented machine code
  - system calls
- virtual machine code
  - interpreted machine
  - JVM
  - portability
  - code size (register assignment)

Output representations

- assembler, for later processing
- relocatable binary (machine instructions)
  - modular compilation
- absolute binary
Structure of a compiler

Program → scanner → tokens → parser

#2 scanner
#3 parser

#1 symbol table

#5 code generator

AST

ASST

abstract syntax

#4 tree

semantic checker

ASST

symbol table

#5 code generator

code generator

code:

`if (a < 39) 3`

integer literal

identifier

expression

`a`

`39`

`int`

if statement

cond then else

expression

if

Special considerations

1) programming
pass by name (Algol 60)
dynamic-sized arrays
nested name scopes
anonymous functions
first-class functions
iterators (Python generators)
automatic object reclamation

2) computer architecture
how many registers?
cache principle: frequent
operations should be fast.
what operations are expensive?
virtual method dispatch, exceptions
effects of caches + paging
3. Specialty compilers
debugging support
IDE integrated development environment
optimizing compilers
retargetable

Chapter 2: Adding calculator 19

types: integer, float

Keywords: f, i, p

variables: lowercase Roman single letters

Syntax: Context-free grammar
Backus-Naur Form (BNF)
Ambiguity in grammars leads to multiple parse trees:

```
if (a)
  if (b)
    S_2
  else
    S_1

S_1
S_2
```

Scanner:

- translate a stream of characters into a stream of tokens.

Stream:

- `peek (stream)`: next item (no modify)
- `advance (stream)`: next item modifies
- `match (stream, item)`: expects the given item, advances stream, error if not.
token: has a type (like id, int, num) ①
has a semantic value (like 'a', 39)

Choices: reserved words: each one a different type ② call them all id, and distinguish in the semantic value ③ call them all type = reserved

Hard-wired scanner (for ac)
Alternative: generated scanner

Regular expressions:
Adequate for defining token syntax.
Input to automatic scanner generators.

1) any string (including empty)
2) concatenation of 2 reg. exps.
3) alternation of 2 reg. exps.

Example: A*b* alternation
As = A | b A*
Formal-Language Hierarchy

Use
- Tokens/Scans
- Syntax/Parser
- Formalism
  - Regular expressions
  - BNF
  - CS grammar
  - Post productions

Language Type
- Regular
- Context-free
- Context-sensitive
- Typed

Automaton
- Finite-state automaton
- Push-down automaton
- Linear-bounded
- Turing machine

Parser

\[ \text{stream of tokens} \xrightarrow{} \text{parser} \xrightarrow{} \text{abstract syntax tree} \]

Many approaches
- Recursive descent
  - RE(1)
  - LH(3)

Patterns:
- \[ f b i a c = 5 b = c + 32 \ p b \]$ 

Programs:
- Decl:
  - def id
  - Decl Decl

- Structs:
  - def id