CSE443
Compilers

Dr. Carl Alphonce
alphonce@buffalo.edu
343 Davis Hall
Syllabus

- Posted on website
- Academic Integrity
Textbook

Classic text.

You should hang on to this one.
Team formation

- If you have a team, please list members in response to Piazza post.
- Also indicate your choice: C or SML.
Keywords have no inherent meaning.

Program meaning is given by formal semantics.

Compiler must preserve semantics of source program in translation to low level form.
Syntax and semantics

- Syntax: program structure
- Semantics: program meaning
- Semantics are determined (in part) by program structure.
Languages: the Chomsky hierarchy

"On Certain Formal Properties of Grammars" published 1959

- recursively enumerable
- context-sensitive
- context-free
- regular
grammars (generators) and languages

recursively-enumerable language

turing machine

context-sensitive language

linear-bounded automaton

caller-free language

push-down automaton

regular finite-state language

finite-state automaton

the traditional Chomsky hierarchy
grammars (generators) and languages

recursively enumerable language

context-sensitive language

case-free language

regular finite-state language

the traditional Chomsky hierarchy

automata (acceptors)

Turing machine

context-sensitive automaton

linear-bound automaton

push-down automaton

the traditional Chomsky hierarchy

Lexical structure

Syntactic structure
Phases of a compiler

Figure 1.6, page 5 of text

Lexical structure

Syntactic structure

character stream

Lexical Analyzer

token stream

Syntax Analyzer

syntax tree

Semantic Analyzer

syntax tree

Symbol Table

Intermediate Code Generator

intermediate representation

Machine-Independent Code Optimizer

intermediate representation

Code Generator

target-machine code

Machine-Dependent Code Optimizer

target-machine code
Phases of a compiler

Figure 1.6, page 5 of text
Lexical Structure

int main(){
Lexical Structure

```c
int main(){
    character stream
}
```
Lexical Structure

character stream -> token stream

```c
int main()
{
    id("int") id("main") LPAR RPAR LBRACE
}
```
Lexical Structure

tokens

- keywords (e.g. static, for, while, struct)
- operators (e.g. <, >, <=, =, ==, +, -, &, .)
- identifiers (e.g. foo, bar, sum, mystery)
- literals (e.g. -17, 34.52E-45, true, ‘e’, “Serenity”)
- punctuation (e.g. {, }, , (, ), , ;)
meta vs object language

- object language: the language we are describing
- meta language: the language we use to describe the object language
meta vs object language

- use quotes (meta vs 'object')
- punctuation (e.g. `{`, `}`, `(`, `)`, `;`)

- use font or font property (meta vs object)
- punctuation (e.g. `{ , } , ( , ) , ; `)
Formally, a language is a set of strings over some alphabet.

Example: \{00, 01, 10, 11\} is the set of all strings of length 2 over the alphabet \{0, 1\}.

Example: \{00, 11\} is the set of all even parity strings of length 2 over the alphabet \{0, 1\}.
Formally, a grammar is defined by 4 items:

1. $N$, a set of non-terminals
2. $\Sigma$, a set of terminals
3. $P$, a set of productions
4. $S$, a start symbol

$G = (N, \Sigma, P, S)$
languages & grammars

\( N \), a set of non-terminals
\( \Sigma \), a set of terminals (alphabet)

\[ N \cap \Sigma = \{\} \]

\( P \), a set of productions of the form (right linear)

\[ X \to a \]
\[ X \to aY \]
\[ X \to \varepsilon \]

\( X \in N, \ Y \in N, \ a \in \Sigma, \ \varepsilon \) denotes the empty string

\( S \), a start symbol

\( S \in N \)
Given a string $\alpha A$, where $\alpha \in \Sigma^*$ and $A \in N$, and a production $A \rightarrow \beta \in P$, we write $\alpha A \Rightarrow \alpha \beta$ to indicate that $\alpha A$ derives $\alpha \beta$ in one step.

$\Rightarrow^k$ and $\Rightarrow^*$ can be used to indicate $k$ or arbitrarily many derivation steps, respectively.
$L(G)$ is the set of all strings derivable from $G$ starting with the start symbol; i.e. it denotes the language of $G$. 

languages & grammars
Given a grammar G the language it generates, \( L(G) \), is unique.

Given a language L there are many grammars H such that \( L(H) = L \).
Lexical Analysis

- Lexical structure described by regular grammar
- Deterministic finite state machine performs analysis
LANGUAGE operations

If \( L \) and \( M \) are regular, so are:

\[
L \cup M = \{ s \mid s \in L \text{ or } s \in M \} \quad \text{UNION}
\]

\[
LM = \{ st \mid s \in L \text{ and } t \in M \} \quad \text{CONCATENATION}
\]

\[
L^* = \bigcup_{i=0, \infty} L^i \quad \text{KLEENE CLOSURE}
\]

By definition, \( L^0 = \{ \varepsilon \} \)