Grammer-1

COM S 319

Topics

- Basics
  - Some definitions
  - Examples
  - Chomsky Hierarchy (or Types of grammers)
- Regular Grammer
  - regular grammer
  - regular expressions
  - examples grep and javascript
  - implementation
    - finite automaton
    - pseudo code
  - using lex

BASICS – SOME DEFINITIONS

List of Terms

- Symbol
- Alphabet
- Language
- Terminals
- Non Terminals
- Production Rule
- Grammer

Symbol

- A symbol here is the smallest distinguishable element in a written language.
- Example: a is a symbol for english language
- Example: Θ is a symbol for greek language
- Example: ♩is a symbol for written music

Albhabet

- An alphabet is a FINITE set of symbols.
  - Note that it has to be FINITE set.

- Example: { '0', '1'} is an alphabet. It just consists of two symbols.

Language

- A language is a set of strings of symbols from some alphabet.

- The empty set and the set consisting of the empty string are also languages and they are distinct from each other.

Terminals, non-terminals, and production rule.

- Non-Terminals are variables which represent a language.
- Example: SENTENCE → NOUN_PHRASE VERB_PHRASE

- Production rules relate variables or non-terminals recursively in terms of each other and primitive symbols called terminals. They have a LHS and a RHS separated by a →

Grammer

- Grammer is denoted by G = { V, T, P, S} where V and T are finite sets of variables and terminals. They are disjoint. P is a finite set of production rules. S is a special variable called the start symbol.
- Example: G1 = {V, T, P, S} where V = {E}, T = {+, -, (,), id}, S = E, P = rules below
  - E → E + E, E → E - E, E → (E), E → id

Grammer

- A string made up of solely of terminals is in the language defined by the grammer if the string can be derived from the start symbol.
- The string "i + (i + j * i + (i + j))" is in the the language defined by the below grammer.
- Example: G1 = {V, T, P, S} where V = {E}, T = {+, -, (,), id}, S = E, P = rules below
  - E → E + E, E → E - E, E → (E), E → id

Showing that "i + (i + j * i + (i + j))" is in the grammer

- production rule 1: E → E + E,
- production rule 2: E → E * E,
- production rule 3: E → (E),
- production rule 4: E → id

- by rule 1: E + E
- by rule 4: i + E
- by rule 3: i + (E)
- by rule 1: i + (E + E)
- by rule 4: i + (i + E)
- finally we will get the string "i + (i + j * i + (i + j))"

A FEW EXAMPLES OF GRAMMERS

Example1

- V = {S, A, B, C}, T = {a, b, c}
- S→ A
- A → aA
- A → B
- B → bC
- C → cC
- C → ε

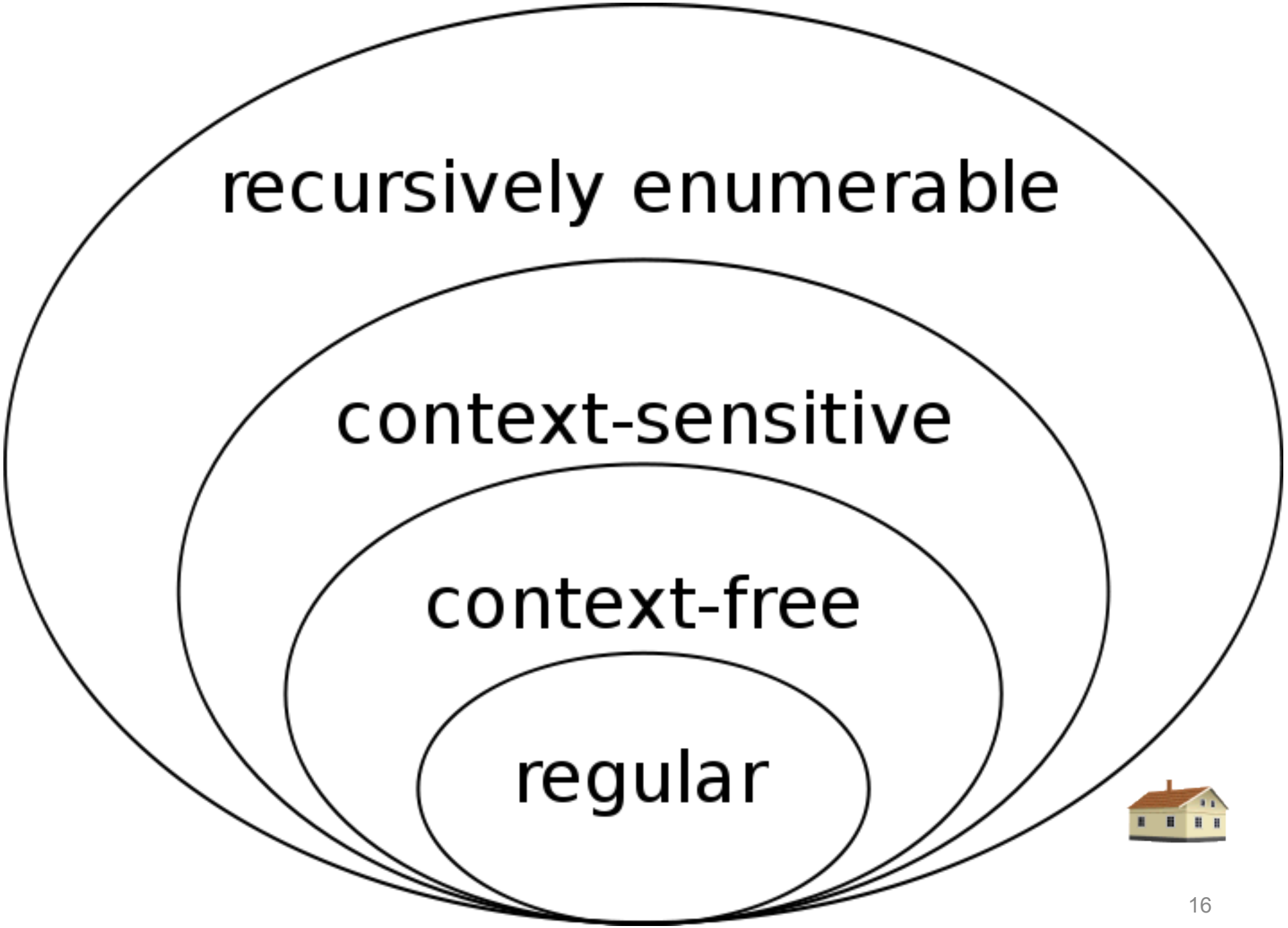
strings in the language?

Example2

- V = {S, A, B, C}, T = {a, b, c}
- S→ aAc
- A → aAc
- A → b

strings in the language?

TYPES OF GRAMMERS
CHOMSKY HIERARCHY


recursively enumerable
context-sensitive
context-free
regular

Chomsky hierarchy

- type-3 or regular grammer
  - can express a^n
  - accepted by finite automaton (limited memory needs)
- type-2 or context-free grammer
  - can express $a^n b^n$ (matching parenthesis, expressions)
  - accepted by pushdown automaton (uses stack)
- type-1 or context-sensitive grammer c
  - an express $a^n b^n c^n$
  - accepted by Linear bounded Turing machines
- type-0 grammers (accepted by Turing Machines)

REGULAR GRAMMER

Regular Grammer

- Production Rules have to be of the form
- $A \rightarrow a$
- $A \rightarrow aB$
- $A \rightarrow \varepsilon$
- where A and B stand for arbitrary variables and a stands for an arbitrary terminal. Epsilon is the empty string.
- There is an equivalent form, where middle rule is $A \rightarrow Ba$

Example1

- This was shown before. This is regular grammer
- V = {S, A, B, C}, T = {a, b, c}
- S→ A
- A → aA
- A → B
- B → bC
- C → cC
- C → ε

strings in the language?

REGULAR EXPRESSION

regular expression

R is a regular expression (is a set of strings) if R is

1. empty set
2. {epsilon} i.e. empty string
3. {a}, for some a in the alphabet
4. R1 U R2, where R1 and R2 are regular expressions
5. R1 o R2 (concatenation)
6. R1* which stands for 0 or more concatenations of R1

regular expression

- . matches any character
- [ ] matches a single character contained in brackets
- [^] matches a single character that is NOT in the brackets
- ^ starting position, $ end position
- * zero or more times
- + one or more times

- Regular expressions express strings in regular language
- Regular grammer also expresses strings in regular language.
- Finite automaton is used to recognize regular expressions

REGULAR EXPRESSION EXAMPLES VIM, GREP, JS

Examples

- In vim, /a[bC].* etc
- grep "a[bC].*] a.txt

- Javascript
  - var myPattern = /a[bC].*/;
  - or var myPattern = new RegExp(s);
  - var match = myPattern.exec(s);
  - var isFound = myPattern.text(s);

REG EXP IMPLEMENTATION

Non-deterministic Finite automaton

- A NFA is a 5-tuple (Q, A, T, S, F)
  - Q is a FINITE set of states
  - A is the alphabet
  - T is the transition function
    - Q x A+ε → P(Q) (i.e. state & alphabet gives state
  - S is the start state
  - F is set of final states
- NFA (non-deterministic finite automaton) can transition to multiple states on the same input and can also transition of epsilon
- easier to express in NFA vs DFA

- Regular expressions, regular grammers, and finite automaton are equivalent.

- a*bc* can be accepted by

- S0 is start state
- state S0 <epsilon, S1>,
- state S1 <a, S1>, <b, S2>,
- state S2 <c, S2>
- S2 is final state

- can use transition table to represent finite automaton
- can use simple traversal over input and keeping future states for implementation. when string has been fully read – is any of the next states a final state? If so accept – else reject.

LEX (LEXER OR LEXICAL ANALYSER)

lex

- describe rules
- lex automatically creates lexical analyzer.
- Example lex rules file:

- %{
- #include <stdio.h>
- %}

- %%
- [a-zA-Z][a-zA-Z0-9]* printf("WORD ");
- [a-zA-Z0-9\/.-]+ printf("FILENAME ");
- \" printf("QUOTE ");
- \{ printf("OBRACE ");
- \} printf("EBRACE ");
- ; printf("SEMICOLON ");
- \n printf("\n");
- [\t]+ /* ignore whitespace */;
- %%

format of lex file

{definitions}

%%

{rules}

%%

{user subroutines}

SUMMARY

- Basics definitions (terminal, alphabet, rules,..
- Chomsky Hierarchy (or Types of grammers)
- Regular Grammer
- Regular expressions
- Examples of RE i n vim, grep, and javascript
- Equivalence of RE, RG, FA
- Implementation NFA using transition table
- Implementation using lex