Course 6
Regular Expressions

The structure and the content of the lecture is based on http://www.eecs.wsu.edu/~ananth/CptS317/Lectures/index.htm

Applications of Regular Expressions

- Unix environments heavily use regular expressions
  - E.g., bash shell, grep, vi & other editors, sed
- Perl scripting – good for string processing
- Lexical analyzers such as Lex or Flex
- Pattern matching (detection of DoS Vulnerabilities in Java Programs - http://www.cs.utexas.edu/~marijn/publications/evil-regexes.pdf, Web programming, Programming web interfaces)

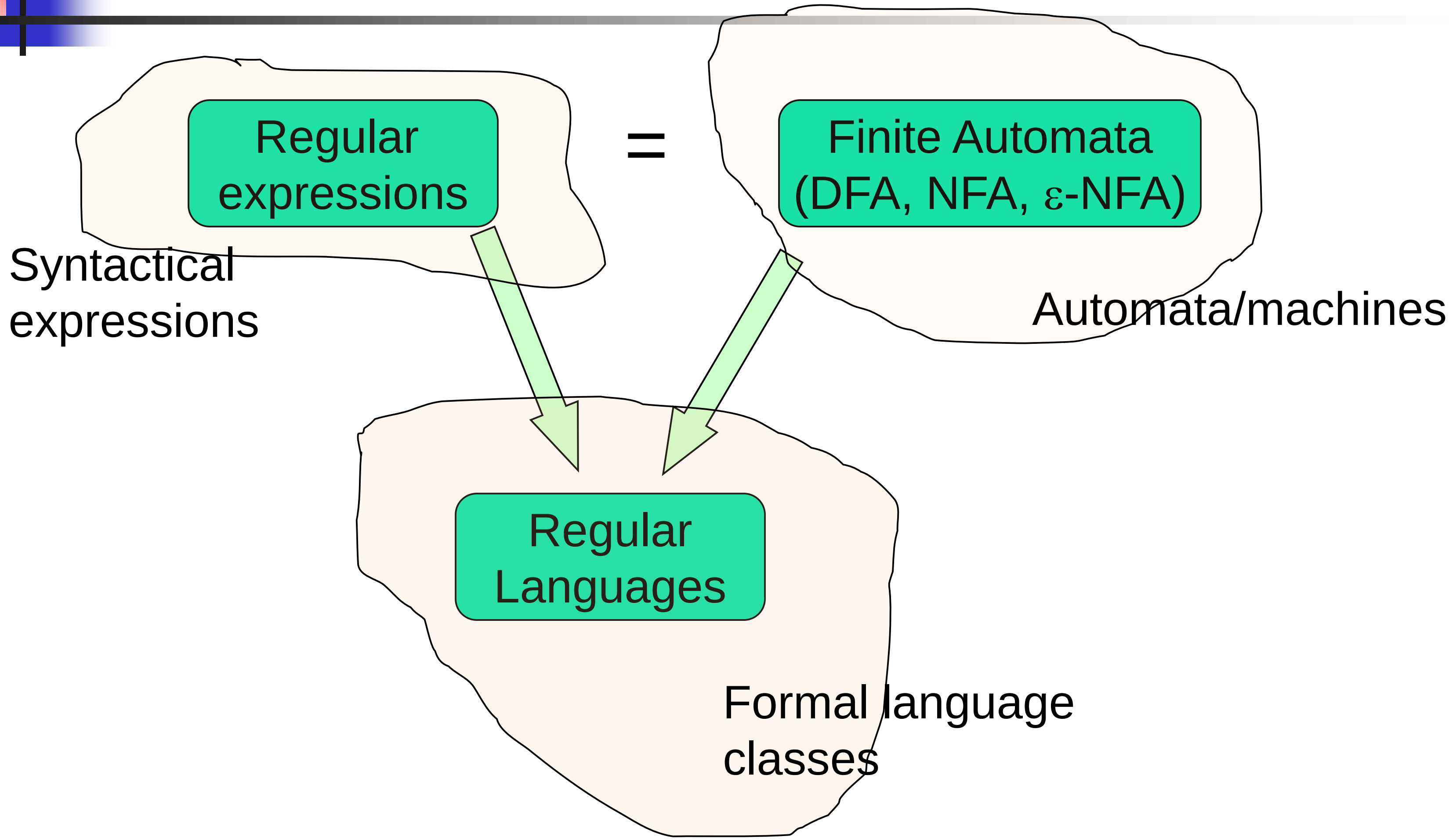
Regular Expressions vs. Finite Automata

- Offers a declarative way to express the pattern of any string we want to accept
  - E.g., 01*+ 10*

- Automata => more machine-like

  < input: string , output: [accept/reject] >
- Regular expressions => more program syntax-like

Regular Expressions



Regular Expressions

Let V - alphabet.

The *regular expressions (r.e)* are words over the alphabet $V \cup \{\bullet, *, |\} \cup \{(,), \emptyset\}$.

Symbols $\bullet$, $*$, $|$ are considered operators.

Note: The following are equivalent: (1) • and . ; (2) +, U and |.

R.e. are inductivelly defined as:

1. λ and $\emptyset$ are r.e.;
2. for all $a \in V$, the word a is r.e.;
3. if R and S are r.e., then $R|S$, $R \bullet S$, R^* are r.e.;
4. any r.e. is built by applying the rules (1)-(3) finitely many times.

Language Operators

- Union of two languages:
  - **L U M** = all strings that are either in L or M ($L|M$)
  - Note: A union of two languages produces a third language
- Concatenation of two languages:
  - **L . M** = all strings that are of the form *xy or* $x \bullet y$ s.t., x $\in$ L and y $\in$ M
  - The *dot* operator is usually omitted
    - i.e., **LM** is same as L.M

"i" here refers to how many strings to concatenate from the parent language L to produce strings in the language L^i

Kleene Closure (the * operator)

- Kleene Closure of a given language L:
  - $L^0 = \{\varepsilon\}$
  - $L^1 = \{w \mid \text{for some } w \in L\}$
  - $L^2 = \{ w_1w_2 \mid w_1 \in L, w_2 \in L \text{ (duplicates allowed)}\}$
  - $L^i = \{ w_1w_2\ldots w_i \mid \text{all } w\text{'s chosen are} \in L \text{ (duplicates allowed)}\}$
  - (Note: the choice of each w_i is independent)
  - $L^* = \bigcup_{i\geq 0} L^i$ (arbitrary number of concatenations)

Example:

- Let L = { 1, 00}
  - $L^0 = \{\varepsilon\}$
  - $L^1 = \{1,00\}$
  - $L^2 = \{11,100,001,0000\}$
  - $L^3 = \{111,1100,1001,10000,000000,00001,00100,0011\}$
  - $L^* = L^0 \cup L^1 \cup L^2 \cup \ldots$

Building Regular Expressions

- Let E be a regular expression and the language represented by E is L(E)
- Then:
  - (E) = E
  - L(E + F) = L(E) U L(F) (+ can be replaced with |)
  - L(E F) = L(E) L(F) (the operator between E,F, resp L(E), L(F) can be replaced with •)
  - L(E*) = (L(E))*

Simplification of regular expressions

- $\alpha + (\beta + \delta) \equiv (\alpha + \beta) + \delta$
- $\alpha + \beta \equiv \beta + \alpha$
- $\alpha + \emptyset \equiv \alpha$
- $\alpha + \alpha \equiv \alpha$
- $\alpha(\beta\delta) \equiv (\alpha\beta)\delta$
- $\alpha\lambda \equiv \lambda\alpha \equiv \alpha$
- $\alpha(\beta + \delta) \equiv \alpha\beta + \alpha\delta$
- $(\alpha + \beta)\delta \equiv \alpha\delta + \beta\delta$
- $\alpha\emptyset \equiv \emptyset\alpha \equiv \emptyset$
- $\lambda + \alpha\alpha^* \equiv \alpha^*$
- $\lambda + \alpha^*\alpha \equiv \alpha^*$
- ...

Building Regular Expressions

- Let E be a regular expression (r.e.) and the language represented by E is L(E)
- Then:
  - (E) = E
  - L(E + F) = L(E) U L(F)
  - L(E F) = L(E) L(F)
  - L(E*) = (L(E))*

Examples

1. $R = (a|b)$. Then $\mathrm{L}(R) = \{a\} \cup \{b\} = \{a, b\}$
2. $R = (ab)$. Then $\mathrm{L}(R) = \{ab\}$
3. $R = a(b|a)$. Then $\mathrm{L}(R) = a\{b, a\} = \{ab, aa\}$
4. $R = a^*$. Then $\mathrm{L}(R) = \bigcup_{j=0}^{\infty}\{a^j\} = \{\lambda, a, a^2, \ldots\} = \{w \in \{a^*\}\} = \{a^n | n \geq 0\}$
5. $R = (a|b)^*$. Then $\mathrm{L}(R) = (L_a \cup L_b)^* = (\{a\} \cup \{b\})^* = (\{a, b\})^*$
6. $R = a(a|b)^*$. Then $\mathrm{L}(R) = a(\{a, b\})^* = \{aw | w \in \{a, b\}^*\}$
7. $R = (b|a)^* a$. Then $\mathrm{L}(R) = (\{b, a\})^* a = \{wa | w \in \{a, b\}^*\}$
8. $R = a(a|b|c)^* c$. Then $\mathrm{L}(R) = \{awc | w \in \{a, b, c\}^*\}$
9. $R = a(a|b)(a|b|c)^*$. Then $\mathrm{L}(R) = \{aaw, abw | w \in \{a, b, c\}^*\} = \{w \in$

Examples

1. $\mathrm{L(R)} = \{w \text{ ends with } b \text{ and contains at least one } a\}. R = (a|b)^* a(a|b)^* b$
2. $\mathrm{L(R)} = \{w \text{ word of } a's \text{ and } b's \text{ odd length}\}. R = ((a|b)(a|b))^*$
3. $\mathrm{L(R)} = \{w \text{ word of } a's \text{ and } b's \text{ with even number of } b's\}.\ R = a^*(a^*ba^*ba^*)^*$
4. $\mathrm{L(R)} = \{w \in \{a, b\}, w \text{ ends with } aa \text{ or } bb\}.\ R = (a|b)^*(aa|bb)$
5. $\mathrm{L(R)} = \{w \in \{1,0\}^*, w \text{ has alternating } 0's \text{ and } 1's\}.\ R = (10)^*|(01)^*|0(10)^*|1(01)^*$
6. $\mathrm{L(R)} = \{w \in \{a, b\}, |w| \equiv 1 \bmod 4\}.\ R = \big((a|b)(a|b)(a|b)(a|b)\big)^*(a|b)$

Precedence of Operators

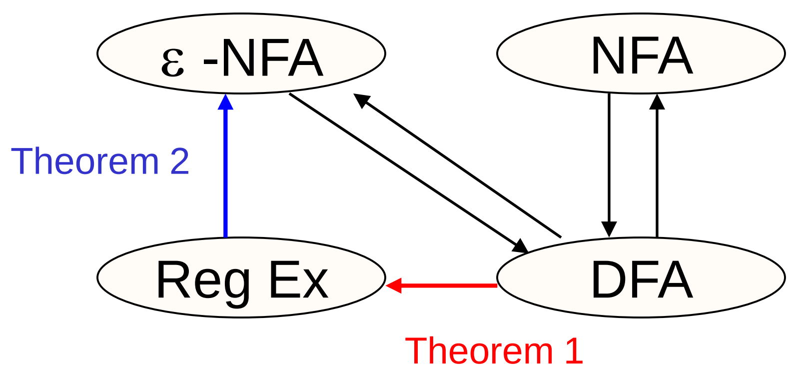
- Highest to lowest
  - * operator (star)
  - **.** (concatenation)
  - + operator

- Example:
  - 01* + 1 = (0 . ((1)*)) + 1

Finite Automata (FA) & Regular Expressions

- To show that they are interchangeable, consider the following theorems:
  - *Kleene Theorem part 1: For every DFA A there exists a r.e. R such that L(R)=L(A).*
  - *Kleene Theorem part 2: For every r. e. R there exists an ε -NFA E such that L(E)=L(R).*

Proofs in the book *Introduction to Automata Theory Languages and Computation by Hopcrof, Motwani, Ullman*

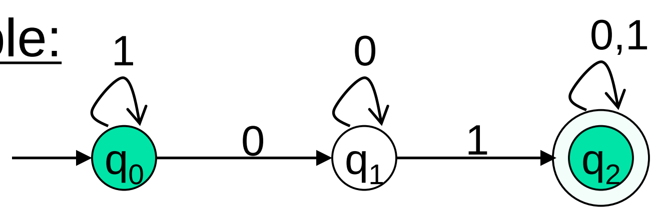


Kleene Theorem

DFA to RE construction

Informally, trace all distinct paths (traversing cycles only once)
from the start state to *each of the* final states
and enumerate all the expressions along the way

Example:



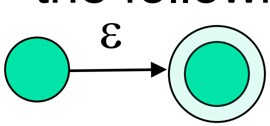
$$\underbrace{(1^*)}_{1^*}\ \underbrace{0\ \ (0^*)}_{00^*}\ \underbrace{1}_{1}\ \underbrace{(0+1)^*}_{(0+1)^*}$$

$$1^*00^*1(0+1)^*$$

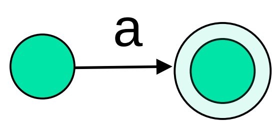
Q) What is the language?

RE to ε-NFA construction

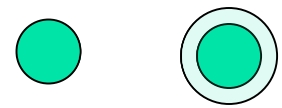
- Given a r.e., we can always built an ε-NFA recognizing L(r.e.) using the following diagrams.



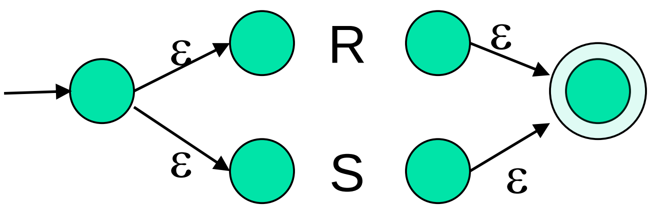
ε-NFA recogn. lang. ε



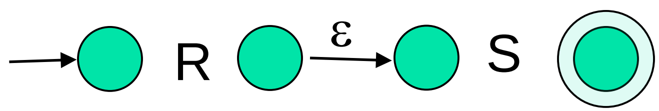
ε-NFA recogn. word. a



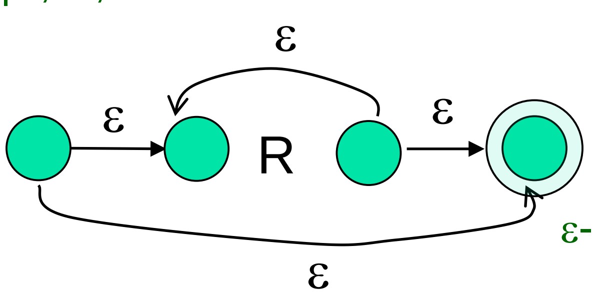
ε-NFA recogn. lang. ∅



ε-NFA recogn. lang. R|S; R,S – r.e.



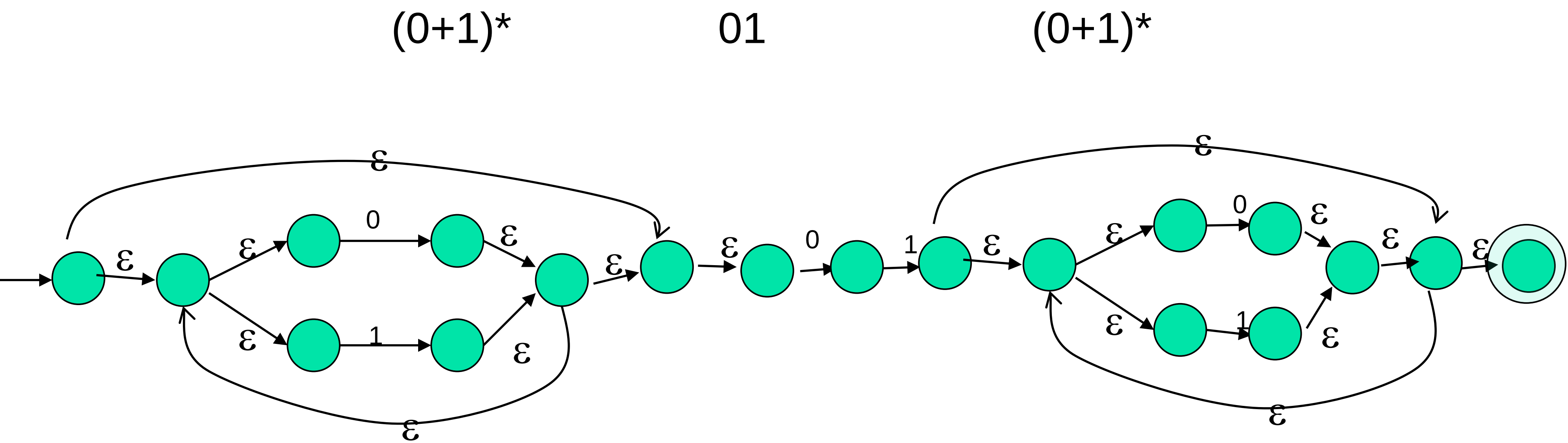
ε-NFA recogn. lang. RS; R,S – r.e.



ε-NFA recogn. lang. R*; R– r.e.

RE to ε-NFA construction

Example: $(0+1)^*01(0+1)^*$



Other examples

Construct ε-NFA for the following r.e.:

- a|b|c
- io|ma
- (a*b)|c*
- (a|b)b*

(see whiteboard)

Summary

- Regular expressions
- Equivalence to finite automata
- DFA to regular expression conversion
- Regular expression to ε-NFA conversion