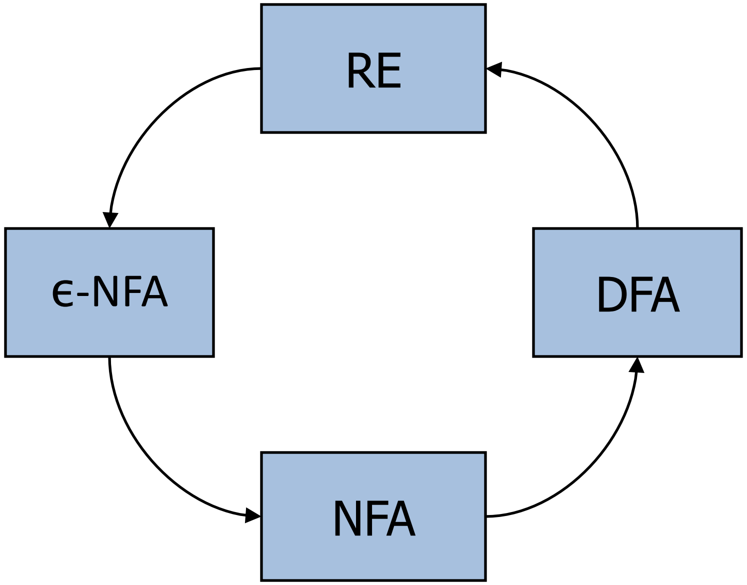
Regular Expressions, Regular Sets and Identities

N Geetha
AM & CS
PSG Tech

What if the Regular Language Is not Represented by a DFA?

- There is a circle of conversions from one form to another:



Regular Expressions

- Useful for representing certain sets of strings in an algebraic fashion
- Used in search commands for finding strings in web browsers / text formatted systems
- Used in Lexical Analyzers to break the source program into logical units called tokens

Regular Expressions vs. Finite Automata

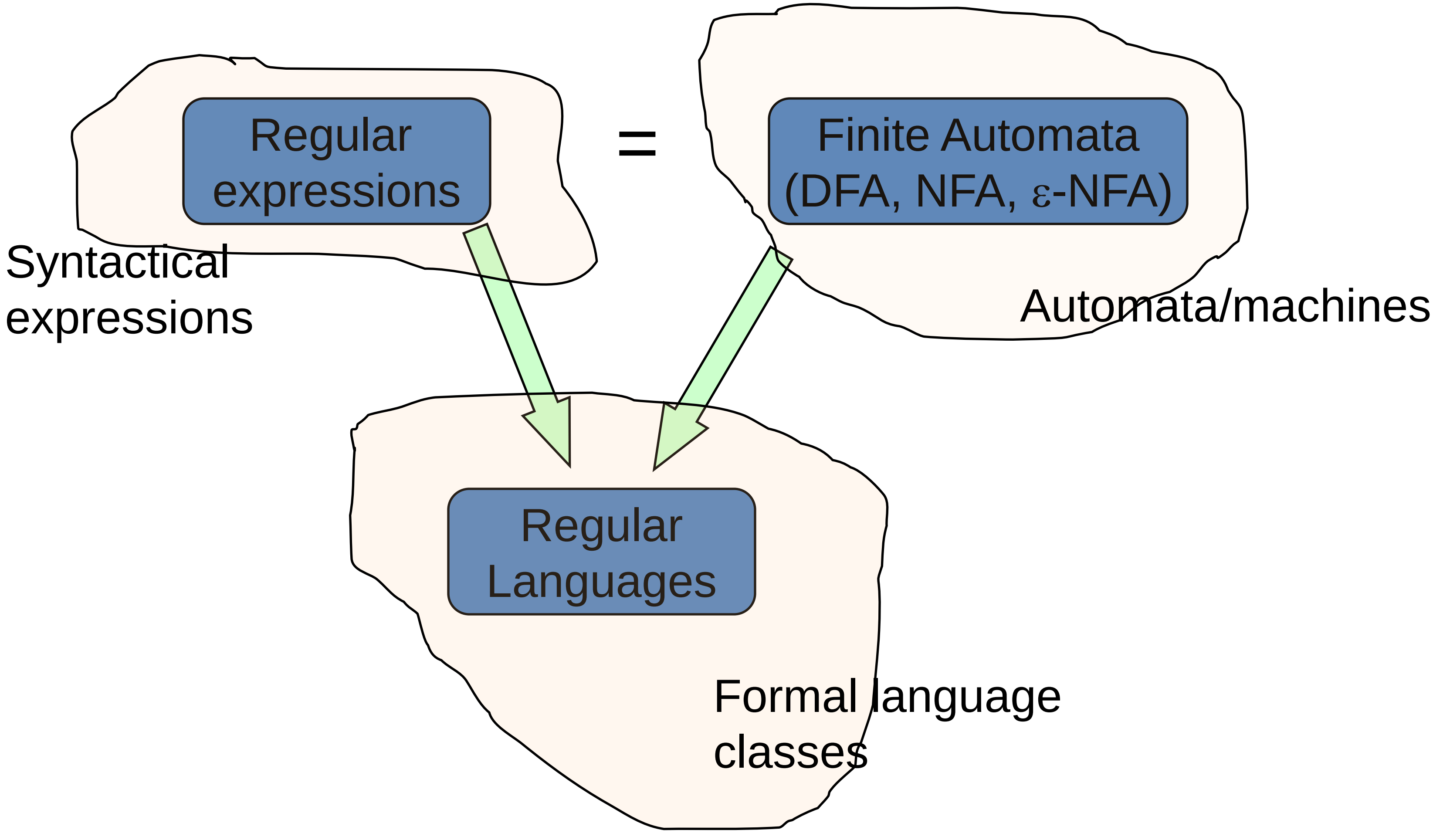
- Offers a declarative way to express the pattern of any string we want to accept
  - E.g., 01*+ 10*

- Automata => more machine-like

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

- 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

Regular Expressions



Regular Expression (RE) Formal Definition

- Basis:
  - single character, a, is an RE, signifying language {a}.
  - ε is an RE, signifying language $\{\varepsilon\}$
  - Ø is an RE, signifying language Ø
- If E_1 and E_2 are REs, then E_1+E_2 is an RE, signifying $L(E_1)$ U $L(E_2)$
- If E_1 and E_2 are REs, then $E_1.E_2$ is an RE, signifying $L(E_1)\ L(E_2)$, that is, concatenation
- If E is an RE, then E* is an RE, signifying L(E)*, that is, Kleene closure, which is the concatenation of 0 or more strings from L(E).
- If E is an RE, then (E) is an RE.
- Parentheses can be used for grouping and don't count as characters.

Precedence of Operators

- Highest to lowest
  - * operator (star)
  - . (concatenation)
  - + operator
- Example:
  - 01* + 1 = (0 . ((1)*)) + 1

Regular Expression Examples

- 1.0* : 1 followed by any number of 0s
- (1.0)* : any number of 10
- 0+0.1 : string 0 or string 01
- 0.(0+1)* : any string beginning with 0
- (0*.1)* : any string not ending with a 0

Regular Set

- Any set represented by an RE is called a regular set.
- Let a,b ϵ ∑;
- a : {a}
- a+b : {a,b}
- a.b : {ab}
- a* : {λ, a, aa, aaa, aaaa, ……}
- (a+b)* : {a,b}*
- The class of regular sets over ∑ is closed under union, concatenation and closure

Regular Set to RE Examples

- {101} :
- {abba} :
- {01,10} :
- {λ ,ab} :
- {abb,a,b,bba} :
- {λ ,0, 00, 000, 0000, ... } :
- {1, 11, 111, 1111, ... }

Regular Set to RE Examples

- Set of all strings of 0s and 1s ending in 00
- Set of all strings of 0s and 1s beginning with a 0 and ending with a 1
- Set of all strings over {a,b} containing exactly 2a’s
- Set of all strings over {a,b} containing atleast 2 a’s
- Set of all strings over {a,b} containing at most 2 a’s
- Set of all strings over {a,b} containing the substring ‘aa’

Regular Language to RE Examples

- $L = \{a^m b^n c^p \ / \ m,n,p >= 1\}$
- $L = \{a^m b^{2n} c^{3p} \ / \ m,n,p >= 1\}$
- $L = \{a^n b a^{2m} b^2 \ / \ m,n >= 1\}$

- $L((a+b)^*(a+bb))$ of length < 4
- $L((aa)^*(bb)^*b)$ of length < 4
- $L((ab+a)^*(aa+b))$ of length < 5

Regular Set Identities

Identities for Regular Expressions.

$P = Q$ if P and Q represent the same set of strings.

I_1 $\Phi + R = R$

I_2 $\Phi R = R.\Phi = \Phi\cancel{R}$

I_3 $\lambda.R = R.\lambda = R$

I_4 $\lambda^* = \lambda$ and $\Phi^* = \lambda$

I_5 $R + R = R$

I_6 $R^* R^* = R^*$

I_7 $RR^* = R^* R$

I_8 $(R^*)^* = R^*$

I_9 $\lambda + RR^* = R^* = \lambda + R^* R$

I_{10} $(PQ)^* P = P.(QP)^*$

I_{11} $(P+Q)^* = (P^* . Q^*)^* = (P^* + Q^*)^*$

I_{12} $(P+Q).R = P.R + Q.R$ and

$R.(P+Q) = R.P + R.Q$.

- are useful for simplifying regul expressions.

Identities

- Are used to simplify REs and for comparison
- Eg.
- $R = (1+011)^*$
- $S= \lambda + 1^*(011)^*(1^*(011)^*)^*$
- $S = \lambda + 1^*(011)^*(1^*(011)^*)^*$
- $= (1^*(011)^*)^*$ Identity 9
- $= (1 + (011))^*$ Identity 11
- $= R$