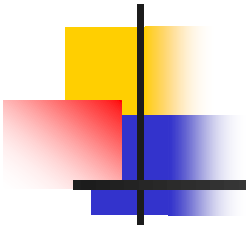
CSC 2204 Finite Automata Theory and Formal Languages

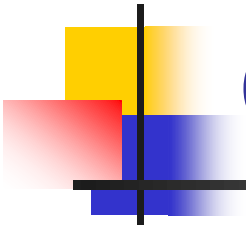
Department of Computer Science
SZABIST (Islamabad Campus)

Week 4 (Lecture 2)



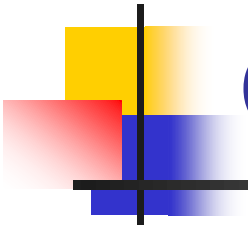
Regular Expressions (REs)

- Can define exactly the same languages that the various forms of automata describe.
- Offer something that automata do not.
  - A declarative way to express the strings we want to accept.
- Serve as the input language for many systems that process string.



Operations on Languages

The *union* of two languages L and M, denoted $L \cup M$, is the set of strings that are in either L or M, or both. For example, if $L = \{001, 10, 111\}$ and $M = \{\epsilon, 001\}$, then $L \cup M = \{\epsilon, 10, 001, 111\}$.



Operations on Languages

The *concatenation* of languages L and M is the set of strings that can be formed by taking any string in L and concatenating it with any string in M.

We denote concatenation of languages either with a dot or with no operator at all, although the concatenation operator is frequently called "dot." For example, if $L = \{001, 10, 111\}$ and $M = \{\epsilon, 001\}$, then $L.M$, or just LM, is $\{001, 10, 111, 001001, 10001, 111001\}$.

Operations on Languages

The *closure* (or *star*, or *Kleene closure*) of a language L is denoted L^* and represents the set of those strings that can be formed by taking any number of strings from L, possibly with repetitions (i.e., the same string may be selected more than once) and concatenating all of them. For instance, if $L = \{0, 1\}$, then L^* is all strings of 0's and 1's. If $L = \{0, 11\}$, then L^* consists of those strings of 0's and 1's such that the 1's come in pairs, e.g., 011, 11110, and ϵ, but not 01011 or 101. More formally, L^* is the infinite union $\cup_{i \geq 0} L^i$, where $L^0 = \{\epsilon\}$, $L^1 = L$, and L^i, for $i > 1$ is $LL \cdots L$ (the concatenation of i copies of L).

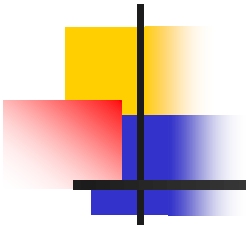
- Exercise: L = {0,11}

Building REs

Use of the Star Operator

Suppose L is the language containing strings of length 1, and for each symbol a in Σ there is a string a in L. Then, although L and Σ "look" the same, they are of different types; L is a set of strings, and Σ is a set of symbols. On the other hand, L^* denotes the same language as Σ^*.

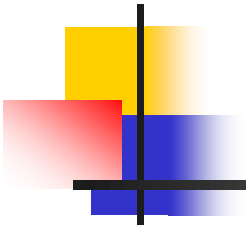
- Example:
  - L = {a,b}, L* = ?
  - ∑ = {a,b}, ∑ * = ?



Regular Expressions

Expressions and Their Languages

Strictly speaking, a regular expression E is just an expression, not a language. We should use $L(E)$ when we want to refer to the language that E denotes. However, it is common usage to refer to say "E" when we really mean "$L(E)$." We shall use this convention as long as it is clear we are talking about a language and not about a regular expression.



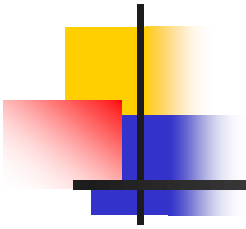
Regular Expressions

BASIS: The basis consists of three parts:

1. The constants ϵ and $\emptyset$ are regular expressions, denoting the languages $\{\epsilon\}$ and $\emptyset$, respectively. That is, $L(\epsilon) = \{\epsilon\}$, and $L(\emptyset) = \emptyset$.

2. If a is any symbol, then $\mathbf{a}$ is a regular expression. This expression denotes the language $\{a\}$. That is, $L(\mathbf{a}) = \{a\}$. Note that we use boldface font to denote an expression corresponding to a symbol. The correspondence, e.g. that $\mathbf{a}$ refers to a, should be obvious.

3. A variable, usually capitalized and italic such as L, is a variable, representing any language.



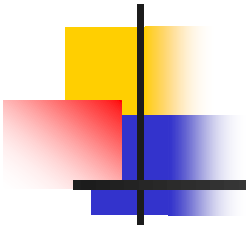
Regular Expressions

INDUCTION: There are four parts to the inductive step, one for each of the three operators and one for the introduction of parentheses.

1. If E and F are regular expressions, then $E + F$ is a regular expression denoting the union of $L(E)$ and $L(F)$. That is, $L(E+F) = L(E) \cup L(F)$.

2. If E and F are regular expressions, then EF is a regular expression denoting the concatenation of $L(E)$ and $L(F)$. That is, $L(EF) = L(E)L(F)$.

Note that the dot can optionally be used to denote the concatenation operator, either as an operation on languages or as the operator in a regular expression.



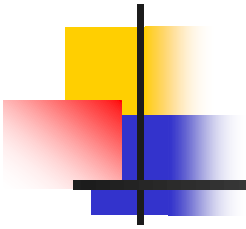
Regular Expressions

3. If E is a regular expression, then E^* is a regular expression, denoting the closure of $L(E)$. That is, $L(E^*) = \big(L(E)\big)^*$.

4. If E is a regular expression, then (E), a parenthesized E, is also a regular expression, denoting the same language as E. Formally; $L\big((E)\big) = L(E)$.

Building REs

- Example: RE for the set of strings that consist of alternating 0s and 1s.
  - REs: 0, 1
  - REs: 01, (01)* **Not Complete, WHY?**
  - REs: 10, (10)* **Not Complete, WHY?**
  - RE: (01)* + (10)* **Not Complete, WHY?**
  - RE: 1(01)* **Not Complete, WHY?**
  - RE: 0(10)* **Not Complete, WHY?**
  - RE: (01)* + (10)* + 1(01)* + 0(10)*
  - RE: (ε+1)(01)*(ε+0) **Alternate RE**



Precedence of RE Operators

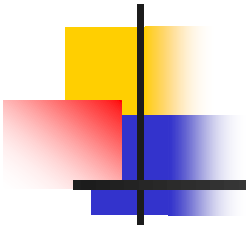
1. The star operator is of highest precedence. That is, it applies only to the smallest sequence of symbols to its left that is a well-formed regular expression.

2. Next in precedence comes the concatenation or "dot" operator. After grouping all stars to their operands, we group concatenation operators to their operands. That is, all expressions that are *juxtaposed* (adjacent, with no intervening operator) are grouped together. Since concatenation is an associative operator it does not matter in what order we group consecutive concatenations, although if there is a choice to be made, you should group them from the left. For instance, **012** is grouped (**01**)**2**.

Precedence of RE Operators

3. Finally, all unions (+ operators) are grouped with their operands. Since union is also associative, it again matters little in which order consecutive unions are grouped, but we shall assume grouping from the left.

- Example: 01* + 1 is (0(1*)) + 1
- Exercise: Build the RE for the set of strings over alphabets {a,b,c} containing at least one a and at least one b.



Regular Expressions

- Exercises:
  - Build REs:
    - Words have at least one a and at least one b.
    - Words have at least two a_s.
  - Give English description:
    - ab*a
    - a*b*
    - (a+c)b*
    - (a+b)*a(a+b)*
    - (a+b)*a(a+b)* + (a+b)*b(a+b)*

Languages and REs

Rule 1 The language associated with the regular expression that is just a single letter is that one-letter word alone and the language associated with $\mathbf{\Lambda}$ is just $\{\mathbf{\Lambda}\}$, a one-word language.

Rule 2 If $\mathbf{r}_1$ is a regular expression associated with the language L_1 and $\mathbf{r}_2$ is a regular expression associated with the language L_2, then:

(i) The regular expression $(\mathbf{r}_1)(\mathbf{r}_2)$ is associated with the product $L_1 L_2$ that is the language L_1 times L_2:

$$\text{language}(\mathbf{r}_1 \mathbf{r}_2) = L_1 L_2$$

(ii) The regular expression $\mathbf{r}_1 + \mathbf{r}_2$ is associated with the language formed by the union of the sets L_1 and L_2:

$$\text{language}(\mathbf{r}_1 + \mathbf{r}_2) = L_1 + L_2$$

(iii) The language associated with the regular expression $(\mathbf{r}_1)^*$ is L_1^*, the Kleene closure of the set L_1 as a set of words:

$$\text{language}(\mathbf{r}_1^*) = L_1^*$$