String v is a prefix of w if w= v y for some string y.

String v is a suffix of w if w= x v for some string x.

String v is a substring of w if there are strings x and y such that w= x v y.

The reversal of string w, denoted w^R, is w "spelled backwards".

Problem of the Day: For w= abaa, what are its

(a) prefixes,

(b) suffixes,

(c) substrings, and

(d) what is w^R?

Announcements

Assignment #1 is due at the beginning of class this Fri. May 25. Hand it in on paper (not electronically).

Make sure you sign the class attendance sheet each class. If you are late, please sign it when you enter the room.

If you want more time for our proof of the day, the notes are usually posted some time the night before or come to class early.

Monday is a holiday (Victoria Day).

Assignment #1 is due next Friday.

Any questions?

Operations on Languages:

1. Complement of L defined over Σ = $\overline{L}$

   $= \{ w \in \Sigma^* : w \text{ is not in } L \}$

2. Concatenation of Languages $L_1 \cdot L_2 = L_1 L_2 =$

   $\{w = x \cdot y \text{ for some } x \in L_1 \text{ and } y \in L_2\}$

3. Kleene star of L, $L^* = \{ w = w_1 w_2 w_3 \dots w_k$ for some $k \geq 0$ and $w_1, w_2, w_3, \dots, w_k$ are all in L}

4. $L^+ = L \cdot L^*$

(Concatenate together one or more strings from L.)

Matrix multiplication:

$$\begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 3 \\ 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 1 & 3 \end{bmatrix}$$

Concatenation:

$ab \cdot bb = abbb$

$bb \cdot ab = bbab$

Σ^* = set of all strings over alphabet Σ

Language over Σ - any subset of Σ^*

Examples: $\Sigma = \{0, 1\}$

$L_1 = \{ w \in \Sigma^* : w$ has an even number of 0's$\}$

$L_2 = \{ w \in \Sigma^* : w$ is the binary representation of a prime number with no leading zeroes$\}$

$L_3 = \Sigma^*$

$L_4 = \{ \} = \Phi$

$L_5 = \{ \varepsilon \}$

$L_2 = \{w \in \{0,1\}^* :$ w is the binary representation of a prime with no leading zeroes$\}$

The complement is:

$\{w \in \{0,1\}^* :$ w is the binary representation of a number which is not prime which has no leading 0's or w starts with 0$\}$

Note: 1 is not prime or composite. The string 1 is in the complement since it is not in L.

$L_1 = \{a, ab\}$
$L_2 = \{ab, b, bb\}$

What are

(a) $L_1 \cup L_2$?
(b) $L_1 \cap L_2$?
(c) $L_1 \cdot L_2$?
(d) $L_1 - L_2$?
(e) L_1^*?
(f) $\Phi \cdot L_2$?
(g) $\{\varepsilon\} \cdot L_2$?

Describe using set descriptor notation, describe the complements of these languages:

(a) L_a = $\{ \varepsilon, a, aa, aaa\}$ over $\Sigma = \{a\}$
(b) L_b = $\{ \varepsilon, a, aa, aaa\}$ over $\Sigma = \{a,b\}$
(c) L_c = $\{0,1\}^* \{0010\} \{0,1\}^*$ over $\Sigma=\{0,1\}$
(d) L_d = $\{001\} \{0,1\}^*$ over $\Sigma=\{0,1\}$
(e) L_e = $\{0,1\}^* \{1101\}$ over $\Sigma=\{0,1\}$

Regular Languages over Alphabet Σ:

[Basis] 1. Φ and $\{\sigma\}$ for each $\sigma \in \Sigma$ are regular languages.

[Inductive step] If L_1 and L_2 are regular languages, then so are:

2. $L_1 \cdot L_2$,

3. $L_1 \cup L_2$, and

4. $L_1{}^*$.

Regular expressions over Σ:

[Basis] 1. Φ and σ for each $\sigma \in \Sigma$ are regular expressions.

[Inductive step] If α and β are regular expressions, then so are:

2. $(\alpha\beta)$

3. $(\alpha \cup \beta)$ and

4. α^*

Note: Regular expressions are strings over

$\Sigma \cup \{ \, (\, , \,) \, , \, \Phi \, , \cup \, , \, * \, \}$

for some alphabet Σ.

Precedence of Operators

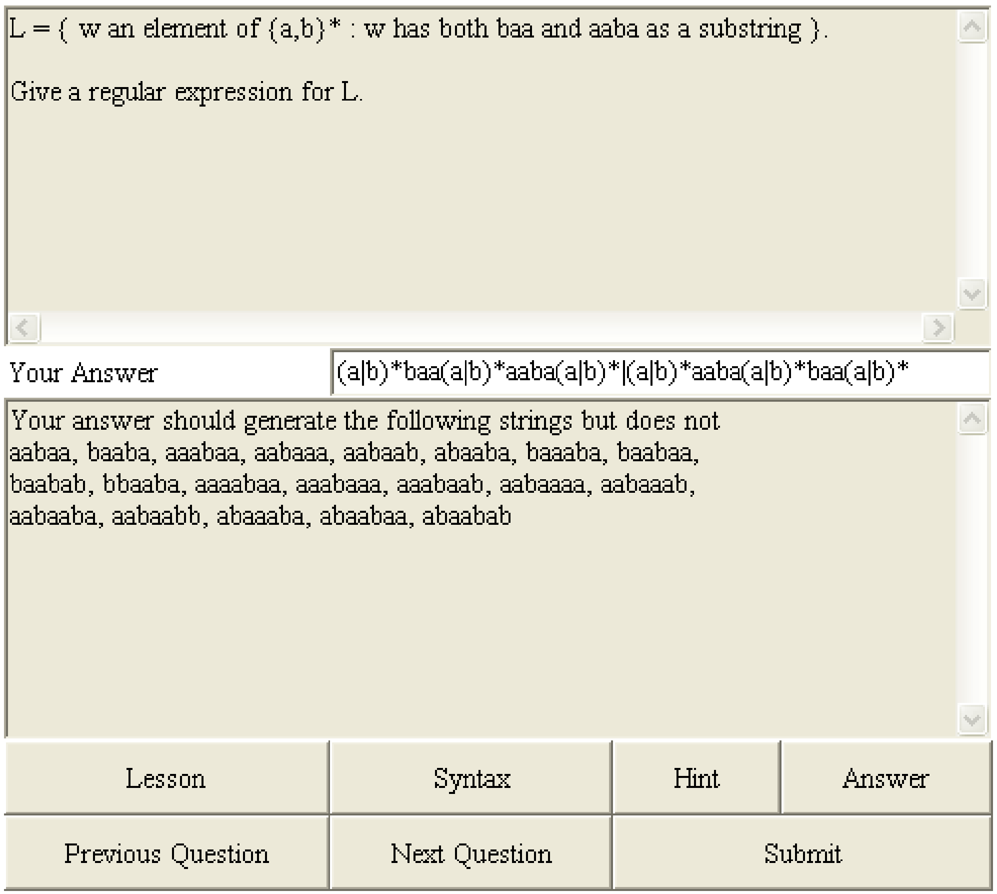
Exponents	highest	Kleene star
Multiplication	⇩	Concatenation
Addition	lowest	Union

Prove the following languages over $\Sigma=\{0,1\}$ are regular by giving regular expressions for them:

1. {w: w has odd length}
2. {w: w contains 0011}
3. {w: w does not contain 01}
4. {w: w starts and ends with the same symbol ($|w| \geq 1$)}

TUTORIAL: L = { w an element of {a,b}* : w has both baa and aaba as a substring }.

(a|b)*baa(a|b)*aaba(a|b)*|(a|b)*aaba(a|b)*baa(a|b)*



MISSING:
aabaa,
baaba,
...

See home page for link to regular expression tutorial