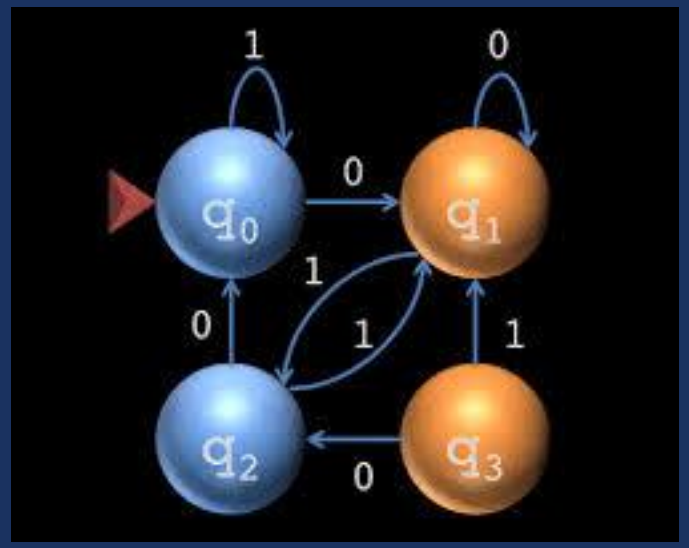
CS3005 –THEORY OF AUTOMATA

LECTURE 2: INTRODUCTION TO FORMAL LANGUAGES & FINITE THEORY OF AUTOMATA

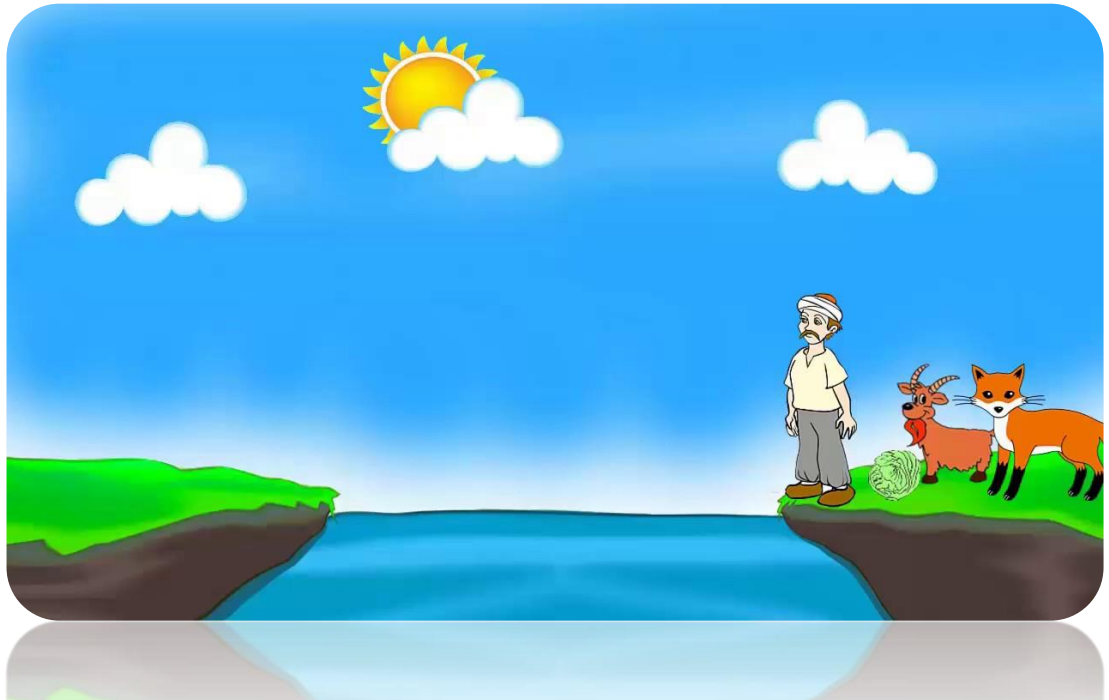


Syed Faisal Ali
sfaisal@nu.edu.pk
Computer Science 4E & 4F

SPRING 2025

PREVIOUS LECTURE - MAN, GOAT, WOLF AND CABBAGE

- Now consider a riddle in which we there is a man has a goat, a wolf, and a cabbage. He has to go across the river by boat but he can only take one thing at a time. If he takes the cabbage but leaves the goat and the wolf together , the wolf will eat the goat. If he takes the wolf but leaves the goat and cabbage together, then the goat will eat the cabbage. How can he get the goat, the wolf and the cabbage across the river?



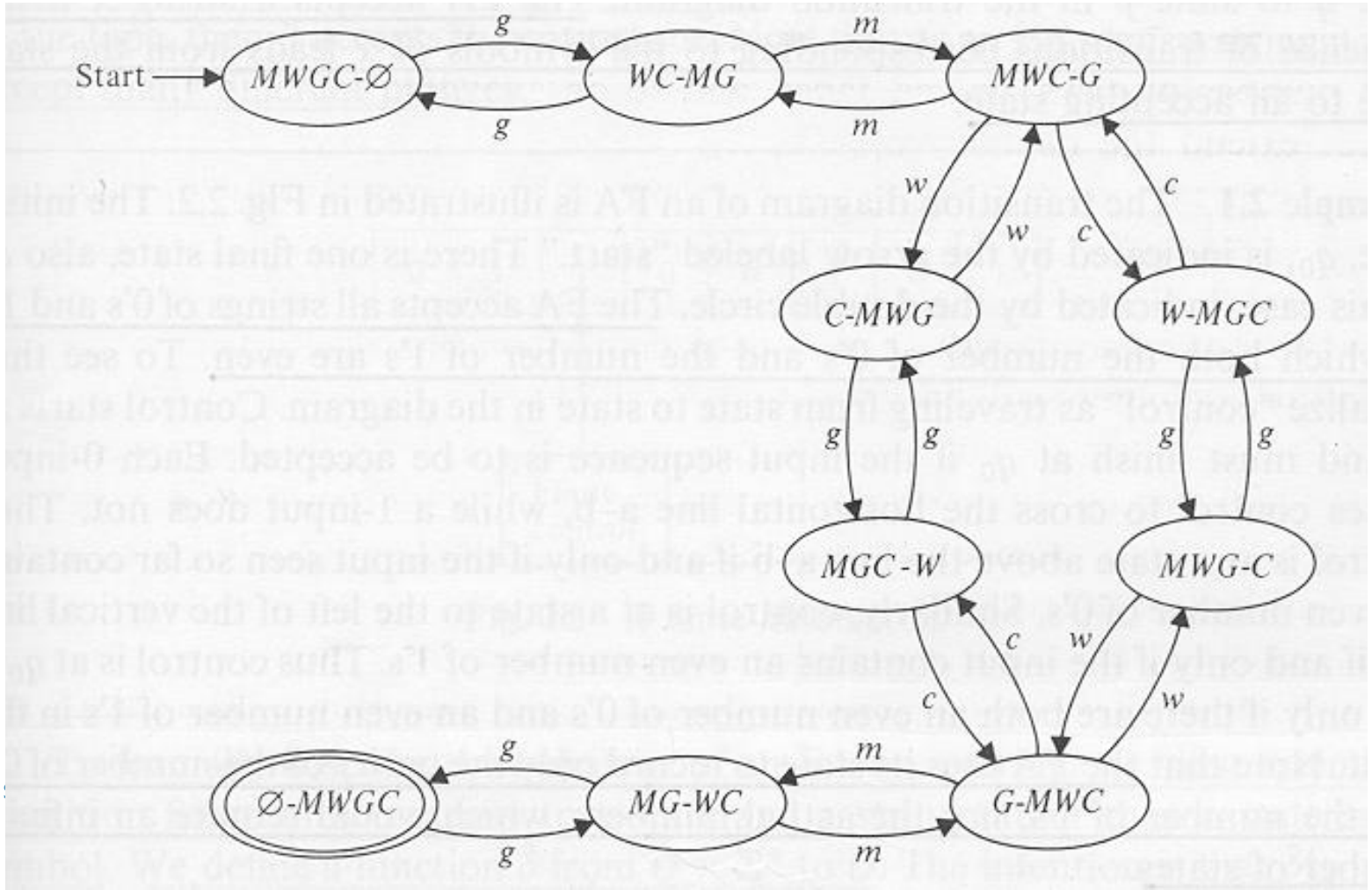
SOLUTION

- The man first takes the goat across the river. He leaves the goat and returns to get the wolf. He leaves the wolf on the other side to but brings back the goat. He then takes the cabbage and leaves the goat. When he is on the other side he reaves the cabbage with wolf. The wolf will not eat the cabbage. The man goes back to pick up the goat and finally returns to the other side where the wolf and cabbage are waiting.
- Now the question is how you are going to solve this riddle using automata?

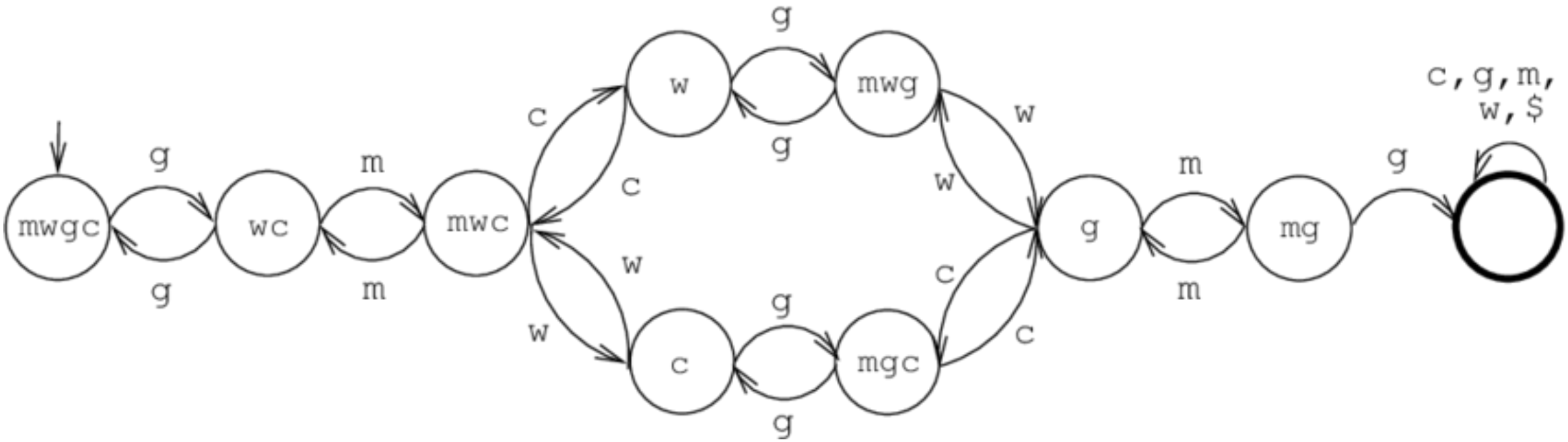
SOLVE THIS RIDDLE USING AUTOMATA

- Step 1 : Find all the elements which are in the riddle.
- Step 2: Find the power set for the riddle.
- Step 3: Find all the constraints in your riddle.
- Step 4: Now identify your initial and final conditions.
- Step 6: Starting from initial condition try to find how we are going to connect to next step.
- Step 7: Continue step 6 until you reach final state.

TRANSITION DIAGRAM FOR MAN, WOLF, GOAT AND CABBAGE



GENERALIZED AUTOMATA FOR THE RIDDLE



LANGUAGES

- $\sum$ = [a b c d e f g h ….z - ’]
- English-Words = {all the words in standard dictionary}
- γ = [the entries in a standard dictionary , plus a blank space, plus the usual punctuation marks]

INTRODUCTION TO DEFINING LANGUAGES

- The set of a language-defining rules can be of two kinds. They can either tell us how to test a string of alphabets that we might be presented with , to see if it is a valid word, or they can tell us how to construct all the words in a language by some clear procedures. We investigate this distinction further in later chapter.
- Let us consider some simple examples of languages. If we start with an alphabet having only one letter, the letter x,
- $\sum$ = [x]
- We can define a language by saying that any non empty string of alphabet characters in a word.
- L_1 ={x xx xxx xxxx …..}
- We could write this in an alternative form :
  - L_1 = {x^n for n = 1 2 3 ….}

LANGUAGE

- In this language, as in any other, we can define the operation of concatenation, in which two strings are written down side by side to form a new longer string. In this example, when we concatenate the word xxx with the word xx, we obtain the word xxxxx.
- The words in this language are clearly analogous to the positive integers, and the operation of concatenation is analogous to addition:
- $x^n\ concatenated\ with\ x^m$ is the word x^{n+m}
- For example a = xxx and b = xx
- Then ab = xxxxx

LANGUAGES

Example Let L_1 = {a, b} and L_2= {c, d}.

- $L_1 . L_2$ = {ac, ad, bc, bd}
- Note: ab differ from ba.

Example ∑= {x}.

- L_1 = {set of all odd words over ∑ with odd length}.
- L_2 = {set of all even words over ∑ with even length}.
- L_1= {x, xxx, xxxxx, xxxxxxx……}.
- L_2= {λ, xx, xxxx, xxxxxx…}.
- $L_1 . L_2$ = {x, xxx, xxxxx, xxxxxxx…}

LANGUAGES

- Consider another language. Let us begin with the alphabet:
- $\sum$ ={*0 1 2 3 4 5 67 8 9*} and define the set of words:
- L_3= { any finite string of alphabet letters that does not start with the letter zero }

LANGUAGE

- This language L_3 then looks like the set of all positive integers written in base 10.

$$L_3 = \{1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ \ldots\}$$

- We say "looks like" instead of "is" because L_3 is only a formal collection of strings of symbols.
- The integers have other mathematical properties.
- If we wanted to define the language L_3 so that it includes the string (word) 0, we could say:
- L_3={ any finite string of alphabet letters that, if it starts with a 0, has no more letters after the first}

LENGTH()

- We define the function "length of a string" to be the number of letters in the string.
- We write this function using the word "length".

For example,

- if a = xxxx in the language L_1 above, then
- length(a) = 4
- If c = 428 in the language L_3 , then length (c) = 3
- If we could directly write that in L_1
- length(xxxx) = 4
- And in L_3, length(428) = 3

LENGTH()

- In any language that includes the empty string λ we have:
- length $(\lambda) = 0$
- For any word w in any language, if length (w) = 0 , then w = λ.

- Another way of defining a language
- L_3 ={*any* finite string of alphabet letters that, if it has length more than one, does not start with a zero}

REVERSE()

- Let us introduce the function reverse. If a is a word in some language L, then reverse(a) is the same string of letters spelled backward, called reverse of a , even of this backward string is not a word in L.

Example

- reverse (xxx) = xxx
- reverse (xxxxx) = xxxxx
- reverse (145) = 541
- Keep in mind that 145 is the word in the language where as 541 is not.

PALINDROME

- Let us define a new language called PALINDROME over the alphabet
- $\sum$ = {a, b}
- Palindrome = {λ , and all the strings x such that reverse(x) = x}

Example:

- Palindrome (145) = FALSE

HOW TO SOLVE PALINDROME?

- Step 1: Get the string S and perform reverse () on it.
- Step 2: Compare original string S with the reversed string R.
- Step 3: If both the results are same then its True and it is Palindrome else not.

Example

- Palindrome (145) = Reverse(145) = 541 , Compare 541 with 145 are they same?? If yes its Palindrome else FALSE it is not.
- Palindrome (101) = Reverse (101) = 101, Compare 101 with 101 are they same?? Yes then its True and it is Palindrome.

KLEENE STAR (*) OR KLEENE CLOSURE

- Given an alphabet ∑, we wish to define a language in which any string of letters from ∑ is a word, even the null string. This language we shall call the closure of the alphabet. It is denoted by writing a star (asterisk) after the name of the alphabet as a superscript.
- ∑* ={ λ, all the members of ∑ with their all combinations and never ended unless will put some restrictions}

KLEENE CLOSURE

- Example:
- If $\sum$ = {x}, then
- $\sum^*$ = L_4 = {λ x xx xxx xxxx …}
- If $\sum$ = L_5 = {0 , 1}
- $\sum^*$ ={ λ 0 00 000 0000 00000 0^n 0^{n+1} 0^{n+2} … 0^m … 0^∞

  1 11 111 1111 11111 1^n 1^{n+1} 1^{n+2} … 1^m … 1^∞

  01 011 0111 01111 011111 01^n 01^{n+1} 01^{n+2} … 01^m … 01^∞ }

  It becomes the superset and comprises of all possible combinations of 0's and 1's.

EXAMPLE

- Example:
- If S = {aa, b}, then
- S* = L_6 = {λ plus any word composed of factors of aa and b}
- = L_6 = {λ plus all strings of a's and b's in which a's occur in even clumps and b occurs in series}
- ={ λ aa aaaa aaaaaa aaaaaaaa $(aa)^n$ $(aa)^{n+1}$ $(aa)^{n+2}$ … $(aa)^m$ … $(aa)^\infty$

  b bb bbb bbbb bbbbb b^n b^{n+1} b^{n+2} … b^m … b^∞

  = aab aabb aabbb aabbbb aabbbbb aab^n aab^{n+1} aab^{n+2} … aab^m … aab^∞ }

It becomes the superset and comprises of all possible combinations of aa's and b's.

Remember: The string aabaaab is not in S*.

EXAMPLE

- Example:
- If S = {a, bb}, then
- S* = L_7 = {λ plus any word composed of factors of aa and b}
- = L_7 = {λ plus all strings of a's and b's in which a's occur in even clumps and b occurs in series}
- ={ λ a aa aaa aaaa $(a)^n$ $(a)^{n+1}$ $(a)^{n+2}$ … $(a)^m$ … $(a)^\infty$

  bb bbbb bbbbbb $(bb)^n$ $(bb)^{n+1}$ $(bb)^{n+2}$ … $(bb)^m$ … $(bb)^\infty$

It becomes the superset and comprises of all possible combinations of a's and bb's.

Remember: The string aabbbaaabb is not in S*.

EXAMPLE

- To prove that a certain word is in the closure language S*, we must show how it can be written as a concatenate of words from the base set S.
- In the last example, to show that abaab is in S* we can factor it as follows:

(ab)(a)(ab)

- These three factors are all in the set S; therefore their concatenation is in S*.
- This is the only way to factor this string into factors of (a) and (ab).
- When this happens, we say that the factoring is unique.
- Sometimes the factoring is not unique.
- For example, consider S = {xx, xxx}

EXAMPLE

- Sometimes the factoring is not unique. For example, consider S = {xx, xxx}.
- Then

$$S^* = \{\lambda \text{ and all strings of more than one x}\}$$

$$= \{(x)^n \text{ for n} = 0, 2, 3, 4, 5, \ldots\}$$

$$= \{\lambda \text{ xx xxx xxxx xxxxx xxxxxx } \ldots\}$$

- Notice that the word x is not in the language S*. The string xxxxxxx is in this closure for any of these three reasons.
- It is :
- (xx)(xx)(xxx) or (xx)(xxx)(xx) or (xxx)(xx)(xx)

EXAMPLE

- If S is the set of three words
- S = $\{w_1\ w_2\ w_3\}$

Then

- $S^+ = \{w_1\ w_2\ w_3 \quad w_1w_1\ \ w_1w_2\ \ w_1w_3\ \ w_2w_1\ \ w_2w_2\ \ w_2w_3$

  $w_3w_1\ \ w_3w_2\ \ w_3w_3\ \ w_1w_1w_1\ \ w_1w_1w_2\ \ldots\}$
- No matter what the words w_1, w_2 and w_3 are.
- If w_1 = aa , w_2 = bbb, w_3 = λ then
- S^+ = {aa bbb λ aaaa aabbb …}

(S*)* = S** = S*

- The words in the set S are listed above in the order corresponding to their w-sequencing, not in the usual size-alphabetical order.
- What happens if we apply the closure operator twice? We start with a set of words S and look at its closure S*.
- Now suppose we start with the set S* and try to form its closure, which we denote as
- (S*)* or S**

THEOREM 1

- For any set S of strings we have S* = S**.
- First let us illustrate what this theorem means.
- Say for example that S = {a,b}.
- Then S* is clearly all strings of the two letters a and b of any finite length whatsoever.
- Now what would it mean to take strings from S* and concatenate them.
- Let us say we concatenated (aaba) and (baaa) and (aaba). The end result (aababaaaaaba) is no more than a concatenation of the letters a and b, just as with all elements of S*.
- aababaaaaaba
- =(aaba)(baaa)(aaba)
- =[(a)(a)(b)(a)] [(b)(a)(a)(a)] [(a)(a)(b)(a)]
- =(a)(a)(b)(a)(b)(a)(a)(a)(a)(a)(b)(a)

THEOREM 1

- Let us consider one more illustration. If S = {aa,bbb}, then S* is the set of all strings where the a's occur in even clumps and the b's in groups of 3, 6, 9,… Some words in S* are
- aabbbaaaa
- bbb
- bbbaa
- If we concatenate these three elements of S* we got one big word in S**, which is again in S*.
- aabbbaaaabbbbbbaa
- [(aa)(bbb)(aa)(aa)][(bbb)][(bbb)(aa)]

PROBLEM TO SOLVE IN CLASS

1. Consider the language S^*, where $S = \{a, b\}$.
   How many words does this language have of length 2? of length 3? of length n?
2. Consider the language S^*, where $S = \{aa, b\}$.
   How many words does this language have of length 4? of length 5? of length 6? What can be said in general?
3. Consider the language S^*, where $S = \{ab, ba\}$. Write out all the words in S^* that have seven or fewer letters. Can any word in this language contain the substrings *aaa* or *bbb*?
4. Consider the language S^*, where $S = \{a\ ab\ ba\}$. Is the string (*abbba*) a word in this language? Write out all the words in this language with seven or fewer letters. What is another way in which to describe the words in this language? Be careful, this is not simply the language of all words without *bbb*.
5. Consider the language S^*, where $S = \{aa\ aba\ baa\}$. Show that the words *aabaa*, *baaabaaa*, and *baaaaababaaaa* are all in this language. Can any word in this language be interpreted as a string of elements from S in two different ways? Can any word in this language have an odd total number of a's?

SOLVE ASSIGNMENT 1

- Solve the problems given on page 23.
- Q 1 to 6, Q9, 12, 13, 14, 15 and 17.
- start doing programming for the functions discussed in the lecture.
- Make a program for calculating the Kleen Operation over alphabet {a, b}

END OF LECTURE