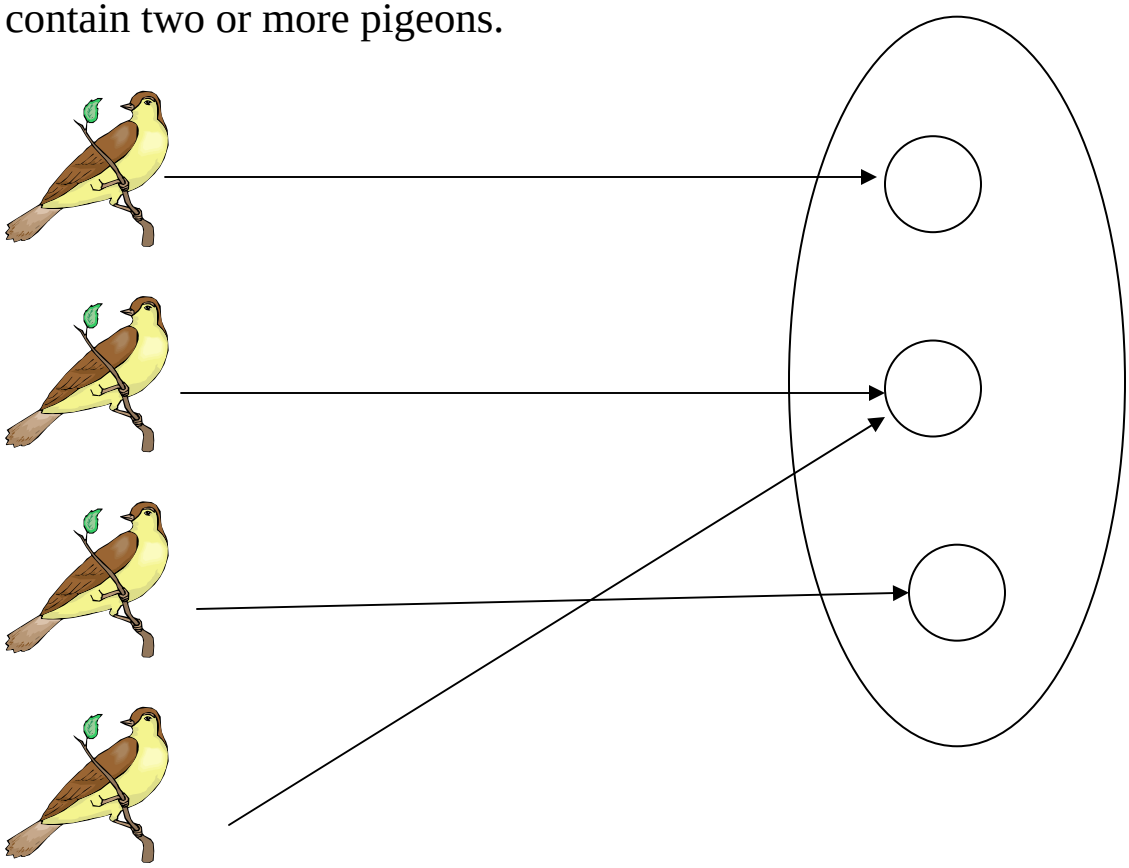
Non Regular Languages

Pigeonhole Principle:

The Pigeonhole Principle states that if n pigeons fly into m pigeonholes and n > m then at least one hole must contain two or more pigeons.



Pigeonhole Principle:
A function from a finite set to a smaller set cannot be one-to-one. There must be at least two elements in the domain that have the same image.

Because a finite state automaton can assume only a finite number of states and because there are infinitely many input sequences, by the pigeonhole principle, there must be at least one state to which the automaton returns over and over again. This is an essential feature of the automaton.
Consider a language that consists of 1 followed by an arbitrary number of 0 and then a 1. Example of inputs strings are 11,101,1001.
Now there are infinitely many such strings and only finitely many states.
Thus, by the pigeonhole principle, there must be a state s_m and two input strings a^p and a^q with $p \neq q$ such that when either a^p or a^q are input to A, A goes to state s_m. (the pigeons are the strings of a's, the pigeonholes are the states and the correspondences associate each string with the state to which A goes when the string is input).

Showing that a language is not regular using the pigeonhole principle:

Example:
$L = \{s \ \sum^* \mid s = a^n b^n \mid n \geq 0\}$
If we attempt to find a DFA that recognizes B, we discover that the machine needs to remember how many a's have been seen so far as it reads the input. So the machine has to keep track of unlimited number of possibilities, this cannot be done with a finite number of states. In this case, we say that the language B is nonregular.
We use the pigeonhole principle to show that B is not regular. We use a proof by contradiction.

Suppose L is regular.
Then some DFA M= (Q, {a,b}, δ, q0, F) exists for it.
Now look at $\delta^*(q_0, a^i)$ for i= 1, 2, .. Since there are an unlimited number of i's, but only a finite number of states in M, the pigeonhole principle tells us that there must be some state, q, such that:
$\delta^*(q_0, a^n)= q$ and $\delta^*(q_0, a^m)= q$ with $m \neq n$.
But since M accepts $a^n b^n$, we must have $\delta^*(q, b^n)=q_f \in F$.

From this we can conclude that

$$\delta^*(q_0,a^m b^n) = \delta^*(\delta^*(q0,a^m), b^n)$$
$$=\delta^*(q, b^n)$$
$$=q_f$$

This contradicts the original assumption that M accepts $a^m b^n$ only if m= n and leads us to conclude that L cannot be regular.

In order to use this type of arguments in a variety of situations, we usually codify it as a general theorem, known as the pumping lemma.

The Pumping Lemma for regular languages:

Let L be an infinite regular language, then there some positive integer m, such that any $w \in L$ with $|w| \geq m$ can be decomposed as:

1. w= xyz
2. $|y| > 0$
3. $w_i= xy^i z \in L$ for all i=0, 1, 2,…

To paraphrase this, every sufficiently long string in L can be broken into three parts in such a way that an arbitrary number of repetitions of the middle part yields another

string in L. We say that the middle string is pumped, hence the term “pumping” lemma.

Proof:
If L is regular, there exists a dfa that recognizes it. Let such a dfa have the states labeled q_0, q_1, …q_n. Now take a string w in L such that $|w| \geq m = n+1$.
Consider the set of states the automaton goes through as it processes w, say q_0, q_i, q_j,…q_f
Since this sequence is exactly $|w| +1$ entries, at least one state must be repeated and such a repetition must start before the nth move. Thus the sequence must look like:
q_0, q_i, q_j,…,q_r,…q_r,…q_f
indicating there must be substrings x, y, z of w such that
$\delta^*(q_0,x) = q_r$
$\delta^*(q_r, y)= q_r$
$\delta^*(q_r,z)= q_f$
with $|xy| \leq n+1=m$ and $|y| \geq 1$.
From this, it follows that:
$\delta^*(q_0,xz) = q_f$
$\delta^*(q_0, xy^2z)= q_f$
$\delta^*(q_0,xy^3z)= q_f$

The pumping lemma is used to show that certain languages are not regular. The demonstration is always by contradiction.

Using the pumping lemma, we show that $L = \{a^nb^n : n\geq 0\}$ is not regular
Assume that L is regular, so that the pumping lemma must hold. We do not know the value of m, but whatever it is, we can always choose n=m.
Therefore, the substring y must consists entirely of a’s. Suppose $|y|= k$. Then the string obtained using i=0 is

$w_0 = a^{m-k}b^m$ and is clearly not in L. This contradicts the pumping lemma and thereby indicates that the assumption that L is regular must be false.

This means that after p a's have been input, at which point A is in state s_m, inputting q additional b's sends A into an accept state, s_q.

In applying the pumping lemma, we must keep in mind what the theorem says. We are guaranteed the existance of an m as well as the decomposition xyz, but we do not know what they are.