Example 4

• Consider $L = \{0^a1^b : a \text{ not equal to } b\}$ where $a, b$ are some integers

• Prove that this is not regular.

• We Prove this by contradiction, using PL4RL

Details of the exponents and conditions

• Number of 0’s cannot be equal to number of 1’s

• This is tricky, since now we have to demonstrate a pumping which gives us a string that has number of 0’s equal to the number of 1’s

• There is a direct proof but it is simpler to use a theorem about regular languages that we plan to prove later on in the course:

Details (cont’d)

• Much easier to prove $\overline{L} = (\Sigma^* - L)$ is not regular

• A property of regular languages: $L$ is regular if and only if $(\Sigma^* - L)$ is regular

• So, if we can prove $\overline{L}$ is not regular, it would be same as proving $L$ is not regular
### Structure of $L_1$

- What does $\overline{L}$ look like?
  - $\Sigma^* - \{0^a1^b: a \neq b\}$
- $\overline{L} = \{0^a1^b: a = b\}$ U
  - \{all strings not of the form $0^a1^b$\}
- Note that for contradicting PL4RL, we can pick any string from $L_1$, so we will pick a string of the form $0^r1^r$

### Applying the PL4RL

- If $\overline{L}$ is regular there is an $n$ as in the PL4RL
- Pick $z = 0^n1^n = uvw$, where $|uv| < n$, $|v| \geq 1$ and
  - $uv^kw$ should belong to $\overline{L}$ for all $k$ greater than or equal to zero.

### Pump up

- Now
- $z = 0000\ldots0000111111\ldots111$ $\overrightarrow{n}$ $\overleftarrow{n}$
- So, $uv$ contains only 0's (since $|uv| < n$)
- Therefore, $v$ contains only 0's.

- $uv^2w = 0^{n+|v|}1^n$
- Since, $|v| \geq 1$, $n + |v| \neq n$
- Therefore, $0^{n+|v|}1^n$ belongs to $L$, hence not in $\overline{L}$
- Hence $uv^kw \notin L$ for $k = 2$
- Thus, $\overline{L}$ violates the PL4RL. So $\overline{L}$ cannot be regular
- By the previous discussion, $L$ cannot be regular
Direct proof

- Consider \( L = \{0^a1^b: a \neq b\} \) where \( a, b \) are some integers.
- Consider \( z = 0^n1^n! + n \)
- \( |z| \geq n \) so that by the PL4RL \( z = uvw \), where \( |uv| < n \), \( |v| > 1 \) and \( uv^k w \) is in \( L \) for all \( k \geq 0 \)
- Clearly \( v \) must be in the 0’s
- Let \( k = 1 + (n! / |v|) \)

Example 5

- Consider \( L = \{0^m!: m \text{ any integer}\} \)
- Prove that this is not regular.

- We Prove this by contradiction, using PL4RL
- Recall \( m! = m*(m-1)*(m-2)....3*2*1 \)

Direct Proof - 2

- It is important that \( k \) be an integer and in fact \( 1 + (n! / |v|) \) is an integer, it is equal to \( 1 + n(n-1)(n-2)...(|v|+1)(|v| - 1)...2.1 \)
- Now, what is \( uv^k w \)? It is \( uv^k w = 0^{n+(k-1)|v|}1^{n!+n} \)
  \( = 0^n + n! + n \)  -- the same number of 0’s and 1’s!  This string is not in \( L \), contradicting the PL4RL

Applying the PL4RL

- If \( L \) is regular there is an \( n \) as in the PL4RL
- Pick \( z = 0^n = uvw \) (\( n \) as in PL4RL) where \( |uv| \leq n \), \( |v| > 1 \) and
- \( uv^k w \) should belong to \( L \) for all \( k \) greater than or equal to zero.
Now
\[ z = 0000 \ldots 0000 \]

\[ n! \]

So, \( uv \) contains only 0's.

Therefore, \( v \) contains only 0's.

\[ \text{Pump up} \]

\[ uv^2w = 0^{n!+|v|} \]

Since \( n \geq |v| \geq 1 \),
\[ n! < n! + |v| \leq n! + n < (n+1)! \]

So, \( (n! + |v|) \) cannot be the factorial of any integer, as it lies between \( n! \) and \( (n+1)! \).

Therefore, \( 0^{n!+|v|} \not\in L \)

\[ \text{Conclusion} \]

Hence \( uv^k w \notin L \) for \( k = 2 \)

Thus, \( L \) violates the PL4RL. So \( L \) cannot be regular.

\[ \text{The string } z \]

If \( z = uw_1w_2 \ldots w_nv \in L \), where \( n \) is the number of states in a DFA for \( L \), then there is a path in the DFA from \( q_0 \) to a final state:

\[ q_0 \xrightarrow{u} p_0 \xrightarrow{w_1} p_1 \]

\[ p_0 \xrightarrow{w_2} p_2 \]

\[ p_2 \xrightarrow{w_n} p_n \xrightarrow{v} \]

\[ \ldots \]
How many \( p_i \)?

- Now, even if \( u = \epsilon \) and \( v = \epsilon \), in which case \( q_0 = p_0 \) and \( p_n \) is final, there are \( n + 1 \) states from \( p_0 \) to \( p_n \) inclusive.

\[ q_0 \xrightarrow{u} p_0 \xrightarrow{w_1} p_1 \]
\[ p_n \xrightarrow{v} p_n \]

\[ p_0 = q_0 \]
\[ p_n = p_{n-1} \]

Pigeon-hole principle

- BUT the DFA only has \( n \) states so two of the \( p \)'s are the same.

- Suppose \( p_i = p_j \) for some \( i, j \), with \( 0 < i < j < n \).

\[ q_0 \xrightarrow{u} p_0 \xrightarrow{w_1} p_1 \]
\[ p_n \xrightarrow{v} p_n \]

The loop - I

Concentrate on the loop.

- The path shown from \( q_0 \) to a final state, continues to reach the final state independent of the number of times the loop is traversed.

\[ w_i \]

The loop - II

- Concentrate on the loop.

\[ w_{i+1} \ldots w_j \]
Turing Machines

- It is time to think about Turing machines (Chapter 7 of the textbook)
- Basic definitions

- WEB SITE:  
  http://bingweb.binghamton.edu/~lander/cs573.html
The basic Turing machine-II

- The movements of the head and the changes of state and the content of the tape are all determined by the transition function

\[ \delta(q, X) = (q', Y, D) \]

The basic Turing machine-III

- A Turing machine is a tuple \( M = (Q, \Sigma, \Gamma, q_0, \delta, B, F) \) where \( Q \) is a finite set of states, \( \Sigma \) and \( \Gamma \) are two alphabets with \( \Sigma \subseteq \Gamma \).
- \( \Sigma \) is the input alphabet, \( \Gamma \) is the tape alphabet

The input

- The tape begins with a string from \( \Sigma^* \) on it and the rest of the tape is blank (\( B \)'s)
- If the tape is infinite in ONE direction, the input string is at the left-hand end of the tape

The initial configuration

- The input must be at the left-hand end
The input configuration-II

- \( B \) is the blank symbol. \( B \) is in \( \Gamma \) but \( B \) is NOT in \( \Sigma \), hence \( \Gamma \) is definitely bigger than \( \Sigma \)
- \( q_0 \) is the initial state, the Turing machine always starts in its initial state at the left hand end of the input string on the tape

Transitions

- The transition function is deterministic in the basic machine:
  \[ \delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\} \]
  \[ \delta(q, X) = (q', Y, D) \]
  where \( D \) is the direction of head movement after writing to the tape, so \( D = L \) or \( R \)

Final states

- \( F \) is the set of final states in \( Q \). If the transitions take you to a final state, the input is accepted...
- no matter how much of the input has been visited

Moves

- To define acceptance in a formal way, we need “moves”
- At any instant in the processing of an input the Turing machine has a “state,” which consists of
  - the state of the control
  - the content of the tape
  - the position of the read/write head
- Such a state is called an “instantaneous description” or i.d.
The i.d. of the following configuration is \( X_1 X_2 X_3 \ldots X_r \).

The general form of an i.d. for a machine with a one-way infinite tape is \( uqv \), where:

- \( u \) is the string of symbols starting at the left-hand end of the tape going up to the symbol BEFORE the read/write head.
- \( q \) is the control state.
- \( v \) is the string of symbols starting AT the head up to the last non-blank.

Special cases-I

The i.d. \( q \) indicates that the tape is completely blank and the read/write head is at the left-most position.

Special cases-II

The i.d. \( qX_1 \ldots X_r \) denotes the situation where the read/write head is at the left-most position.

ANY of the \( X_i \) EXCEPT \( X_r \) could be \( B \).
Special case-III

- The i.d. $X_1...X_r q$ denotes the situation where the read/write head is at a blank B and all positions to the right are blank.
- ALL of the $X_i$ could be B

\[ \begin{array}{cccccccc}
X_1 & X_2 & X_3 & . & . & . & X_r & B & B & B & B & B \\
\end{array} \]

A move from one i.d. to another

- A move describes how a transition affects the current i.d. of the machine.
- We use the notation $uqv \|\rightarrow u'q'v'$
- The exact form of the move depends on the transition.

Right moves

- Suppose the direction of the transition is to the Right:
  \[ \delta (q, X) = (q', Y, R) \]
- We have various possibilities:

\[ \begin{array}{cccccccc}
X_1 & X_2 & X_3 & . & . & . & X_r \qquad Yq' Y_1...Y_s \\
\end{array} \]

Most general case of a right move

- If the i.d. is $X_1X_2...X_r q XY_1...Y_s$ and the transition is $\delta (q, X) = (q', Y, R)$, then
  \[ X_1X_2...X_r q XY_1...Y_s \|\rightarrow X_1X_2...X_r Yq' Y_1...Y_s \]
Most general case of a right move - 2

\[
X_1 X_2 \ldots X_r X Y_1 Y_2 \ldots Y_s B
\]

\[\vdash X_r q X Y_1 \ldots\]

\[\vdash \ldots X_r Y q' Y_1 \ldots\]

At left-hand end

- If the i.d. is \(q X Y_1 \ldots Y_s\) and the transition is \(\delta(q, X) = (q', Y, R)\), then \(q X Y_1 \ldots Y_s \vdash Y q' Y_1 \ldots Y_s\)

A move right at left-hand end

\[
X Y_1 Y_2 \ldots Y_s B B B B B
\]

\[\vdash q X Y_1 \ldots Y_s \vdash Y q' Y_1 \ldots Y_s\]

All blanks to the right

- If the i.d. is \(X_1 X_2 \ldots X_r q X\) and the transition is \(\delta(q, X) = (q', Y, R)\), then \(X_1 X_2 \ldots X_r q X \vdash X_1 X_2 \ldots X_r Y q'\)
Blanks at the read/write head and to the right - 2

- If the i.d. is $X_1X_2...X_rq$ and the transition is $\delta(q, B) = (q', Y, R)$, then $X_1X_2...X_rq \rightarrow X_1X_2...X_rYq'$
- (Remember $Y$ may be blank itself)

---

Left moves

- Suppose the direction of the transition is to the Right:
  $\delta(q, X) = (q', Y, L)$
- We have various possibilities but we also have the constraint of not moving past the left-hand end of the tape
General case

- If the i.d. is $X_1X_2\ldots X_r q XY_1\ldots Y_s$ and the transition is $\delta(q, X) = (q', Y, L)$, then $X_1X_2\ldots X_r q XY_1\ldots Y_s$.

At left-hand end

- If the i.d. is $q XY_1\ldots Y_s$ and the transition is $\delta(q, X) = (q', Y, L)$, then there is no move.

All blanks to the right

- If the i.d. is $X_1X_2\ldots X_r q X$ and the transition is $\delta(q, X) = (q', Y, L)$, then $X_1X_2\ldots X_r q X$.

General case - 2

- If the i.d. is $X_1X_2\ldots X_r q XY_1\ldots Y_s$ and the transition is $\delta(q, X) = (q', Y, L)$, then $X_1X_2\ldots q'X_r Y Y_1\ldots Y_s$. 

- If the i.d. is $X_1X_2\ldots X_r q X$ and the transition is $\delta(q, X) = (q', Y, L)$, then $X_1X_2\ldots q'X_r Y$.
If the i.d. is $X_1 X_2 \ldots X_r q$ and the transition is $\delta(q, B) = (q', Y, L)$, then

$X_1 X_2 \ldots X_r q \quad \vdash \quad X_1 X_2 \ldots q' X_r Y$

We extend the move $\vdash$ to $\vdash^*$ much as we did with $\delta$ in the finite automata.

The non-move: $u q v \vdash^* u q v$

We write $u q v \vdash^* u_n q_n v_n$ when there is a sequence if i.d.'s and moves:

$u q v \vdash u_1 q_1 v_1 \vdash u_2 q_2 v_2 \vdash u_3 q_3 v_3 \ldots \vdash u_n q_n v_n$
Acceptance

- The language $L(M)$ accepted by the Turing machine is defined using $|--^*$:

$$L(M) = \{ w \in \Sigma^* : q_0 w |--^* uqv \text{ where } q \text{ is a final state} \}$$

- Notice that there is no requirement that you must reach the end of the input in order to accept a string with a Turing machine

Example: a machine for $(0 + 1)^*$

- A machine to recognize the language $(0 + 1)^*$ can be very simple:

$$M = (\{q_0\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_0\})$$

$\delta$ does not define any transitions

A machine for $(0 + 1)^*$

- A machine to recognize the language $0^*$ can be very simple:

$$M = (\{q_0,q_1\},\{0,1\},\{0,1,B\}, \delta, q_0, B, \{q_1\})$$

$\delta (q_0, 0) = (q_0, 0, R)$

$\delta (q_0, B) = (q_1, B, R)$
A machine for $0^*-\text{II}$

\[ \delta(q_0, 0) = (q_0, 0, R) \]

A machine for $0^*-\text{III}$

\[ \delta(q_0, B) = (q_1, B, R) \]

A machine for \{0^n 1^n : n \geq 0\}

- The machine is described in the textbook
- We can run the simulator on a number of inputs...

Format of the machine description

1. ATM
2. TM for \{0^n 1^n : n \geq 0\}
3. 0 1 // input alphabet, note succ
4. 0 1 X Y B // tape alphabet
5. 1 // number of tapes
6. 1 // number of tracks on tape 0
7. 1 // tape 0 is 1-way infinite
8. q0 // the initial state
9. q4 // final state(s)
Format of the machine description-II

q0  0  q1  X  r  // the transition
q0  Y  q3  Y  r
q0  B  q4  B  r
q1  0  q1  0  r
q1  1  q2  Y  l
q1  Y  q1  Y  r
q2  0  q2  0  l
q2  X  q0  X  r
q2  Y  q2  Y  l
q3  Y  q3  Y  r
q3  B  q4  B  r
end  // required

initial state of simulator

Accept Input Tape  Step  #

0 0 0 0 1 1 1

Current state: q0
Previous state:

<Step  Stop  Step> . . .

write X and move right to find a 1

Accept Input Tape  Step: 4

X 0 0 0 1 1 1

Current state: q1
Previous state: q0

<Step  Stop  Step> . . .

write a Y, move left to find an X

Accept Input Tape  Step: 5

X 0 0 0 1 1 1

Current state: q2
Previous state: q1

<Step  Stop  Step> . . .
find the X, move right and start over

Accept Input Tape  Step: 9

X 0 0 0 Y 1 1 1

Current state: $q_0$
Previous state: $q_2$

write an X and move right, looking for a 1

Accept Input Tape  Step: 10

X X 0 0 Y 1 1 1

Current state: $q_1$
Previous state: $q_0$

found a 1, set it to Y

Accept Input Tape  Step: 13

X X 0 0 Y 1 1 1

Current state: $q_1$
Previous state: $q_1$

moving left to find an X (past Y’s and 0’s)

Accept Input Tape  Step: 14

X X 0 0 Y Y 1 1

Current state: $q_2$
Previous state: $q_1$
after 4 iterations, find X, move right & go to $q_0$

Accept Input Tape  Step: 35

Current state: $q_2$
Previous state: $q_2$

Done with 0's, just look for spare 1's

Accept Input Tape  Step: 36

Current state: $q_0$
Previous state: $q_2$

State $q_3$ can only pass Y's

Accept Input Tape  Step: 37

Current state: $q_3$
Previous state: $q_0$

$\delta (q_3, B) = (q_4, B, R)$ and $q_4$ is final

Accept Input Tape  Step: 41

Current state: $q_4$
Previous state: $q_3$