Every Set can be represented by some pattern.

In Theory of computation, Regular Languages are represented by some Expressions called Regular Expressions.

Regular expressions use limited number of regular operators.

Regular Expressions can be made richer by using any number of logical connectives so they can describe languages in more descriptive and easier way.

This leads to the more complexity for the construction of state diagram from Regular Expressions.

A new concept of Derivatives of Regular Expressions is proposed which leads in a very natural way, to the construction of a state diagram from a Regular Expression containing any number of logical operators.
Regular Expressions

- Regular operators used in Regular Expressions are:
  
  For P and Q, two sets of sequences of symbols...

  - Product or Concatenation Operator
    \[(P \cdot Q) = \{ s | s = pq; \ p \in P, \ q \in Q \}\]

  - Union Operator
    \[(P + Q) = \{ s | s = p \text{ or } s = q; \ p \in P, \ q \in Q \}\]

  - Iterate or Star Operator
    \[P^* = \bigcup P^n \text{ for } n \geq 0\]
Along with these operators Logical operators such as AND, Complement etc can be used in the Regular Expression to make them more easier and descriptive.

For Ex.

Regular Expression R which represent the set of all sequences having three consecutive 1’s but not those ending in 01 or consisting of 1’s only.

\[ R = (1111) \& (I01+11^*) \] where \( I = (0+1)^* \)
Derivatives of Regular Expressions

- Given a set $R$ of sequences and a finite sequence $s$, the derivative of $R$ with respect to $s$ is denoted by $D_s R$ and is $D_s R = \{t \mid st \not\in R\}$.

- For example if $R = ab(ba)^*$ then $D_{ab} R = (ba)^*$
Defining $\delta(R)$

Given any set $R$ of sequences we define $\delta(R)$ to be

$$\delta(R) = \begin{cases} 
\lambda & \text{if } \lambda \in R, \\
\Phi & \text{if } \lambda \notin R.
\end{cases}$$

- $\delta(a) = \Phi$ for any $a \in A_k$, $\delta(\lambda) = \lambda$ and $\delta(\Phi) = \Phi$
- $\delta(P^*) = \lambda$
- $\delta(PQ) = \delta(P) \& \delta(Q)$

If $R = f(P,Q)$ it is also easy to determine $\delta(R)$ as follows

- $\delta(P+Q) = \delta(P) + \delta(Q)$,
- $\delta(P\&Q) = \delta(P) \& \delta(Q)$,
- $\delta(P') = \begin{cases} 
\lambda & \text{if } \delta(P) = \Phi, \\
\Phi & \text{if } \delta(P) = \lambda.
\end{cases}$
If $R$ is regular expression, the derivative of $R$ with respect to a sequence $a$ of unit length ($a \in A_k$) is found recursively as follows:

- $D_a a = \lambda$
- $D_a b = \Phi$, for $b = \lambda$ or $b = \Phi$ or $\Phi A_k$ and $b \neq a$,
- $D_a (P^*) = (D_a P)P^*$,
- $D_a (PQ) = (D_a P)Q + \delta(P)D_a Q$,
- $D_a (f(P,Q)) = f(D_a P, D_a Q)$.
Derivatives of Regular Expressions with respect to larger string

- The derivative of a regular expression $R$ with respect to a finite sequence of symbols $s = a_1a_2 \ldots a_r$ is found recursively as follows:

- $D_{a_1a_2}R = D_{a_2}(D_{a_1}R)$,
- $D_{a_1a_2a_3}R = D_{a_3}(D_{a_1a_2}R)$,
- $D_sR = D_{a_1a_2 \ldots a_r}R = D_{a_r}(D_{a_1a_2 \ldots a_{r-1}}R)$,
- If $s = \lambda$ then $D_\lambda R = R$.
Properties of Derivatives

- The derivative $D_s R$ of any regular expression $R$ with respect to any sequence $s$ is a regular expression.
- If $s=\lambda$ then $D_\lambda R = R$ is Regular.
- For length of $s=1$, for eg $D_a R$ gives expressions which are regular expressions.
- For $s$ is of length greater than 1, $D_s R$ is sequence of $D_a R$ where $a$ is the any unit length symbol of $s$; gives that $D_a R$ is also regular expression.
A sequence $s$ is contained in a regular expression $R$ if and only if $\lambda$ is contained in $D_s R$.

$D_s R = \lambda$ this means $s \lambda \in R$ that is $\lambda \in R$
Every regular expression $R$ can be written in the form

$$R = \delta(R) + \sum a D_a R$$

for $a \in A_k$ where the terms in the sum are disjoint.

For a characteristic derivative $D_s R$, we can write

$$D_s R = \delta D_s R + \sum a D_{sa} R$$

Here $D_{sa} R$ is the next consecutive derivative of $D_s R$.

If $D_{ua} R$ is characteristic derivative equal to $D_s R$ then above equation becomes

$$D_s R = \delta D_s R + \sum a D_{ua} R$$

This is called characteristic equation of $R$.

We can get $R$ after solving this characteristic equation.
A sequence $s$ is accepted by an automaton $M$ with starting state $q_\lambda$ iff when $s$ is applied to $M$ in $q_\lambda$ the output is 1 at the end of $s$. Otherwise $s$ is rejected by $M$. A sequence $s$ is accepted by a state $q_j$ of $M$ iff when $M$ is started in $q_j$ the output is 1 at the end of $s$.

Two states $q_j$ and $q_k$ of $M$ are indistinguishable iff every sequence $s$ applied to $M$ started in $q_j$ produces the same output sequence as that produced by applying $s$ to $M$ started in $q_k$. 
Two states $q_j$ and $q_k$ of $M$ are indistinguishable iff $R_j$ and $R_k$ denoting the sets of sequences accepted by $q_j$ and $q_k$ are equal.

Here $s_j = b$ and $s_k = a$

Then it gives that two states $q_j$ and $q_k$ of an automaton $M$ characterized by the regular expression $R$ are indistinguishable iff their derivatives are equal, i.e. $D_{s_j}R = D_{s_k}R$, where $s_j$ and $s_k$ are any two sequences taking $M$ from state $q_\lambda$ to $q_j$ and $q_k$ respectively.
State Diagram Construction

- For given $A_k = \{0,1\}$ and a regular expression $R$ over it; We can find $D_s R$ where $s = \lambda, 0, 1, 00, 01, 10, 10, 000, 001, \ldots$

- After a certain value of $s$ we will get the same value of $D_s R$ which indicates that $R$ has only a finite number of distinct derivatives.

- This property is used to construct a state diagram.

- Here, state diagrams are considered as Moore machine (outputs are related to states) or Mealy machine (outputs are related to transitions).
To obtain the minimal state diagram from a regular expression, find the characteristic derivatives and associate one internal state with each characteristic derivative. The output associated with a state is $Z = 1$ iff the corresponding characteristic derivative contains $\lambda$. 
For Example $R = (0+1)^*1$

Derivatives of $R$ will be follows:
- $D_\lambda = R$; Introduce $q_\lambda$ with $Z = 0$
- $D_0 = R$; Return to $q_\lambda$ under input 0
- $D_1 = R+\lambda$; Introduce $q_1$ with output $Z = 1$
- $D_{00}$ and $D_{01}$ need not be considered as $D_0$ does not correspond to a new state.
- $D_{10} = R$ Go from $q_1$ to $q_\lambda$ under input 0
- $D_{11} = R+\lambda$ Return from $q_\lambda$ to $q_1$ for input 1
Is the state diagram minimal?

- The state diagram constructed by associating one internal state with each type of derivative will always be minimum if we can recognize the equality of two regular expressions.
- Recognizing the equality of two regular expressions is difficult.
**Similarity**: Two regular expressions are similar if one can be transformed to the other by using only the identities:

- \( R + R = R \),
- \( P + Q = Q + P \),
- \( (P + Q) + R = P + (Q + R) \)

- Every regular expression has only a finite number of dissimilar derivatives.
For Example $R = (I00I) \& (I01)'$ where $I = (0+1)^*$

- $D_\lambda = R = P \& Q' \quad \text{Introduce } q_\lambda$
- $D_0 = (P + 0I) \& (Q + 1)' \quad \text{Introduce } q_0$
- $D_1 = P \& Q' \quad \text{Return to } q_\lambda$
- $D_{00} = (P + 0I + I) \& (Q + 1)' = (Q + 1)' \quad \text{Introduce } q_{00}$
- $D_{01} = P \& (Q + \lambda)' \quad \text{Introduce } q_{01}$
- $D_{000} = (Q + 1)' \quad \text{Return to } q_{00}$
- $D_{001} = (Q + \lambda)' \quad \text{Introduce } q_{001}$
- $D_{010} = (P + 0I) \& (Q + 1)' \quad \text{Return to } q_0$
- $D_{011} = P \& Q' \quad \text{Return to } q_\lambda$
- $D_{0010} = (Q + 1)' \quad \text{Return to } q_{00}$
- $D_{0011} = Q' \quad \text{Introduce } q_{0011}$
- $D_{00110} = (Q + 1)' \quad \text{Return to } q_{00}$
- $D_{00111} = Q' \quad \text{Return to } q_{0011}$
The behavior of a multiple-output sequential circuit can be represented by specifying one regular expression for each output.

The action of a sequential circuit with \( r \) outputs can be described by an ordered \( r \)-tuple of regular expressions, referred as regular vectors, \( R = (R_1, R_2, \ldots, R_r) \).

The derivative of a vector \( R \) of regular expressions, with respect to a sequence \( s \), is a vector of regular expressions denoted by \( D_s R \) and defined by \( D_s R = (D_s R_1, D_s R_2, \ldots, D_s R_r) \).
Regular Expressions for Multiple-Output Circuits

- Two regular vectors $P$ and $Q$ with $r$ components each, are equal, $P = Q$, iff their components are equal, i.e. $P_i = Q_i$ for all $i$.
- Every regular vector has a finite number of types of derivatives.
Theorem

- Given a regular vector $R$, if a state diagram is constructed by associating one type of derivative of $R$ per state, that state diagram represents the desired behavior and is minimal.
Required: \( Z_i = 1 \), as a result of applying a sequence \( s \) iff \( s \in R_i \).
- By construction, when \( s \) is applied, the state diagram will go from the start state \( q \) to \( q_s \), associated with \( D_s R \) or with the equivalent previous derivative.
- Also by construction, the output \( Z_i = 1 \) iff \( \lambda \in D_s R_i \), i.e. \( S \in R_i \).
- So, every sequence will produce the desired output vector.
The state diagram is minimum:

- There must be a distinct state for each distinct derivative

- If \( D_s R \neq D_t R \), then \( D_s R_j \neq D_t R_j \), for some \( j \).

  So, \( D_s R \) and \( D_t R \) cannot correspond to the same state, for the output \( Z_j \) would be incorrect for some sequence which is in \( D_s R_j \) but not in \( D_t R_j \) or vice versa.
Regular Expressions for Multiple-Output Circuits

For Example \( R = (R_1,R_2) = ((0+10*1)*10*1,(0+1)*01) \)

\[
\begin{align*}
D_\lambda &= (R_1,R_2) \quad \text{Introduce } q_\lambda \\
D_0 &= (R_1,R_2+1) \quad q_0, Z = (0,0) \\
D_1 &= (0*1R_1+0*1,R_2) \quad q_1, Z = (0,0) \\
D_{00} &= (R_1,R_2+1) \quad \text{To } q_0, Z = (0,0) \\
D_{01} &= (0*1R_1+0*1,R_2+\lambda) \quad \text{To } q_1, Z = (0,1) \\
D_{10} &= (0*1R_1+0*1,R_2+1) \quad q_{10}, Z = (0,0) \\
D_{11} &= (R_1+\lambda,R_2) \quad \text{To } q_\lambda, Z = (1,0) \\
D_{100} &= (0*1R_1+0*1,R_2+1) \quad \text{To } q_{10}, Z = (0,0) \\
D_{101} &= (R_1+\lambda,R_2+\lambda) \quad \text{To } q_\lambda, Z = (1,1)
\end{align*}
\]
Proof of finiteness of the no. of derivatives of a Regular Expression

- Proof is by induction on a number N of regular operators.

- Basis, N = 0. The theorem is certainly true when R is one of \( \varepsilon \) or \( a \ A_k \), for we have
  - \( D_s \varepsilon = \phi \) for all \( s \in I \),
  - \( D_s a = a \) and \( D_s \varepsilon = \phi \) for all \( s \in I, s \neq a \)
  - \( D_s a = a. D_a a = a. D_s a = a \) for all \( s \in I, s \neq a \), \( a \).

- Thus we have \( d = 1, d = 2 \) and \( d_a = 3 \).
INDUCTION STEP, $N > 0$. Assume that each expression $X$ with $N$ or fewer operators has a finite number $d_X$ of derivatives. If $R$ is an expression with $N + 1$ operators, there are three cases.
Case 1. \( R = f(PQ) \). It is easily verified from the definitions that
\[
DsR = Ds(P + Q) = DsP + DsQ. \quad \text{Thus} \quad d_R \leq d_P \quad d_Q.
\]
If \( R = P' \) then \( DsR = (DsP)' \).

In this case, \( d_R = d_P \). Since any Boolean function can be expressed using a finite number of sums and complements, it follows that the number of derivatives of \( R \) (of the form \( R = f(P, Q) \)) is finite.
Case 2. \( R = PQ \). Let \( s = a_1 a_2 \ldots a_r \).

We have \( D_{a_1} R = (D_{a_1} P)Q \delta + (P)D_{a_1} Q \).

Similarly, for a sequence of length 2, we have
\[
D_{a_1 a_2} R = (D_{a_1 a_2} P)Q \delta + (D_{a_1} P) D_{a_2} Q \delta + (P) D_{a_1 a_2} Q.
\]

In general, the derivative with respect to a sequence of length \( r \) will have the form
\[
D_{a_1 \ldots a_r} R = (D_{a_1 \ldots a_r} P)Q \delta + (D_{a_1 \ldots a_{r-1}} P) D_{a_r} Q \delta + \ldots + (D_{a_1} P) D_{a_2 \ldots a_r} Q + (P) D_{a_1 \ldots a_r} Q.
\]
Thus $D_s R$ is the sum of $(D_s P)Q$ and of at most $r$ derivatives of $Q$. If there are $d_p$ and $d_Q$ types of derivatives of $P$ and $Q$ respectively, there can be at most

$$d_R \leq d_p 2^{d_Q}$$

types of derivatives of $R$.

Hence the inductive step holds for this case also.
Case 3. $R = P^*$ Again let us consider the formation of the derivative of $P^*$.

We have

$$D_{a_1}(P^*) = (D_{a_1}P)P^*,$$

$$D_{a_1a_2}(P^*) = (D_{a_1a_2}P)P^* \delta (D_{a_1}P) D_{a_2}(P^*)$$

$$= (D_{a_1a_2}P)P^* + (D_{a_1}P)(D_{a_2}P)P^*, \text{ etc.}$$

It can be seen that, in general, $D_s R$ will be the sum of terms of the form $D_t(P)R^*$. If $P$ has $d_P$ types of derivatives, then $R$ has at most $d_R d_{2d_P - 1}$ types of derivatives.
Conclusion

1. Regular expressions can be obtained more easily from word description of problems if one is allowed to use any logical connective in the formation of the expression.

2. The concept of derivative of a regular expression is a powerful aid in analyzing the properties of regular expressions with arbitrary logical connectives.

3. The derivative approach leads naturally to state diagrams of sequential circuits and has been extended to cover the multiple output case.
Thank You