Regular Expressions

A sequence of characters that form a search pattern.

How does this relate to regular languages?

Regular expressions are built from the regular operations discussed earlier.

Regular operations

- $\cup$ union
- $\circ$ Concatenation
- $*$ Kleene Star

Regular expressions are a fundamental part of modern computer science.

Tools such as AWK and GREP are based entirely on Regular Expressions

Grep = global regular expression print

Most modern languages (Python, Perl, Ruby) all have implementations of regular expressions

Formal Definition of Regular Expressions

R is a regular expression if R is

1. a for some $a \in \Sigma$
2. ε
3. $\emptyset$
4. $(R_1 \cup R_2)$ where R_1 and R_2 are regular expressions
5. $(R_1 \circ R_2)$ where R_1 and R_2 are regular expressions
6. R_1^*, where R_1 is a regular expression

Called an Inductive definition

Very similar to the definition of a Regular Language.

A language is Regular iff some regular expression describes it.

Proof is in the book

Theorem 1.54 p 66 - 76

Unix regular expressions

a = some single character

a|b = a or b union

ab = concatenation

a^* = kleene star

. = any single character a U b U ... V z ascii is a finite alphabet

a+ = one or more a's aa^*

a? = ~~zero~~ or one a $a \cup \varepsilon$

^a = ~~ends with a~~ starts with a

a$ = ends with a

a{2,} = 2 or more

[abc] = a or b or c

[a-z] = any letter

\d = any number 0-9

\s = any whitespace

All of these can be assembled using just the regular operations.

R is a regular expression if R is

1. a for some $a \in \Sigma$
2. ε
3. $\emptyset$
4. $(R_1 \cup R_2)$ where R_1 and R_2 are regular expressions
5. $(R_1 \circ R_2)$
6. R_1^*

what is the language of the regular expression ε? $\{\varepsilon\}$

Note that R_1 and R_2 are always smaller than R

This makes this an inductive definition and not a circular one.

To distinguish between R and the language recognized by R

$L(R)$ R

Theorem

If a language is described by a Reg Ex then it is regular.

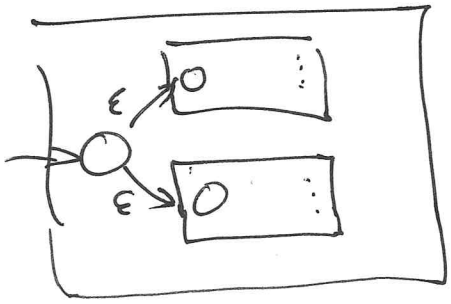
- Create a DFA/NFA for each of the rules that can be used to create a Reg Ex

a

ε

$\emptyset$ 

$R_1 \cup R_2$



$R_1 \circ R_2$

R_1^*

Theorem:

If a language is regular it is described by a RegEx.

Proof:

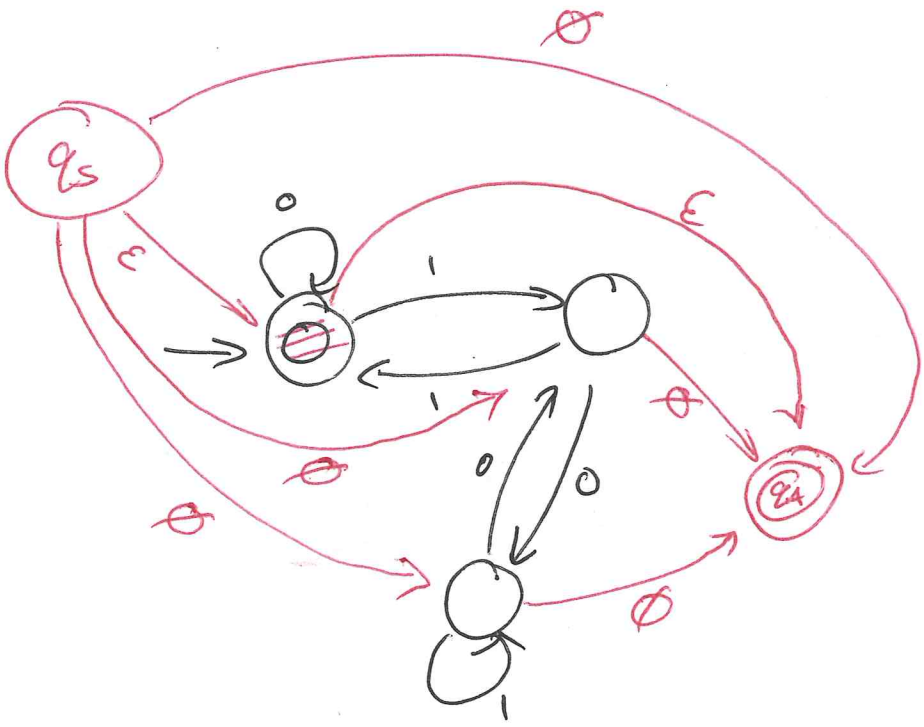
Convert DFA to equivalent regular expressions.

Generalized NFA GNFA

each transition is labelled with a regEx instead of a single symbol

Meets following Conditions

- Start state has a transition to every other state but no incoming transitions.
- Accept state has a transition from every other state but no outbound transitions
- All other states have arrows to each other



With each Step we will reduce the number of states in the GNFA by 1

When the number of states = 2 we're finished

GNFA formal definition

5-tuple $(Q, \Sigma, \delta, q_{start}, q_{accept})$

$$\delta : (Q - \{q_{accept}\}) \times (Q - \{q_{start}\}) \rightarrow R$$

where R is the collection of all regular expressions over the alphabet Σ

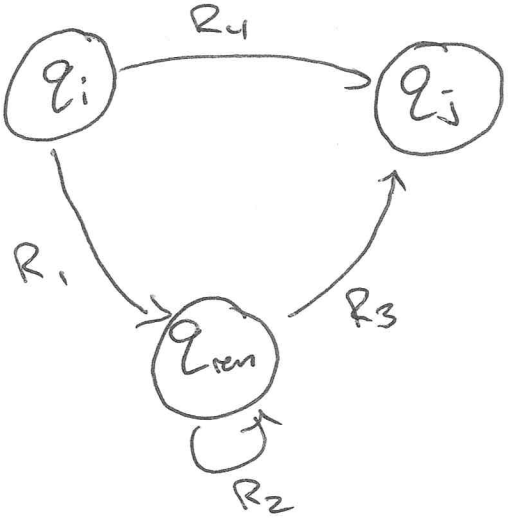
$\delta(q_i, q_j) = R$

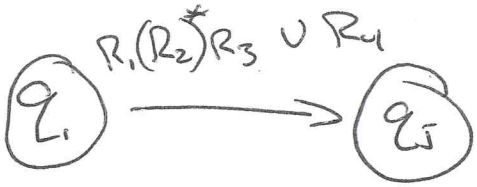
since there is a transition from every state to every state we define the transitions function with the source and destination functions as input and the Reg Ex as output

GNFA computation

A GNFA accepts a string $w \in \Sigma^*$ if $w = w_1 w_2 \ldots w_k$ where each $w_i \in \Sigma^*$ and a sequence of states $q_0, q_1, \ldots q_k$ exists such that

1. $q_0 = q_{start}$
2. $q_k = q_{accept}$
3. For each i, we have $w_i \in L(R_i)$ where $R_i = \delta(q_{i-1}, q_i)$





Convert(G)

1. Let k = number of states in G
2. If $k = 2$, then G must consist of q_{start} and q_{accept} with a single arrow connecting them labelled with R
3. if $k > 2$

   select any state $q_{rem} \in Q$

   $q_{rem} \neq q_{start}$ and $q_{rem} \neq q_{accept}$

   Let $G' = (Q', \Sigma, \delta', q_{start}, q_{accept})$

   $Q' = Q - \{q_{rem}\}$

   and for any $q_i \in Q' - \{q_{accept}\}$ and $q_j \in Q' - \{q_{start}\}$

   $$\delta'(q_i, q_j) = (R_1)(R_2)^*(R_3) \cup (R_4)$$

   $R_1 = \delta(q_i, q_{rem})$, $R_2 = \delta(q_{rem}, q_{rem})$, $R_3 = \delta(q_{rem}, q_j)$, $R_4 = \delta(q_i,$

4. Return Convert(G')

For any GNFA G, convert(G) is equivalent to G.

Base Case: $k = 2$

must have a start state, an accept state and a single arrow labelled with R

Induction Step:

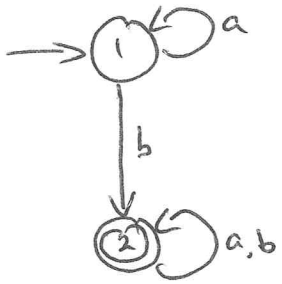
- Assume the statement is true for k-1 states
- For some string w suppose G accepts w

then there exist a sequence of states

$$q_{start}, q_1, q_2, \ldots, q_{accept}$$

if none of those states is q_{rem} then G' also accepts w because the RegEx from q_i to q_i is still part of the resulting union.

DFA



GNFA