Week 6: Lecture Notes

Substitution:

$f : \Sigma \to \Delta^*$, $\Sigma \to$ an alphabet
$\Delta \to$ another alphabet

defined by $f(a) = R_a$, where $R_a \subseteq \Delta^*$ is a regular set for $a \in \Sigma$.

Extending f to strings $f : \Sigma^* \to \Delta^*$

i) $f(\varepsilon) = \varepsilon$

ii) $f(xa) = f(x) f(a)$

Extending f to languages $f : S \to \Delta^*$, $S \subseteq \Sigma^*$

$$f(L) = \bigcup_{x \in L} f(x)$$

Example:

Let $f(0) = a$, $f(1) = b^*$

- Then $f(010) = ab^*a$ defines another regular set.
- if L is the language defined by $0^*(0+1)1^*$

then

$$f(L) = a^*(a+b^*)(b^*)^*$$
$$= a^*(a+b^*)b^*$$
$$= a^*b^*$$

Theorem:

The class of regular sets is closed under substitution.

Proof:

Let $R \subseteq \Sigma^*$ be a regular set.

For each $a \in \Sigma$, let $R_a \subseteq \Delta^*$ be a regular set

Define a substitution mapping

$$f: \Sigma \to \Delta^* \quad \text{as} \quad f(a) = R_a$$

Now consider regular expressions denoting R and each R_a

Replace each symbol a in the regular expression for R by the regular expression for R_a

Then $$f(R) = \bigcup_{a \in R} f(a) = \bigcup_{a \in R} R_a$$

can be proved by using induction on the number of operators in the regular expression.

Note:

- $f(L_1 \cup L_2) = f(L_1) \cup f(L_2)$
- $f(L_1 L_2) = f(L_1) f(L_2)$
- $f(L_1^*) = (f(L_1))^*$

Homomorphisms

A substitution h such that $h(a)$ contains a single string from Δ^* for each a.

Example:

Let $h(0) = aa$, $h(1) = aba$

Then, $h(010) = aaabaaa$

Let $\underline{L_1 = (01)^*}$, then $h(L_1) = (aaaba)^*$

Inverse Homomorphisms of a language L

For a string a $\quad h^{-1}(L) = \{x \mid h(x) \text{ is in } L\}$

$h^{-1}(w) = \{x \mid h(x) = w\}$

Example:

$h(0) = aa$, $h(1) = aba$

$L_2 = (ab+ba)^* a$. Then $h^{-1}(L_2) = ?$

Claim:

$h^{-1}(L_2) = \{1\}$, $h^{-1}(L_2) = \{x \mid h(x) \in L_2\}$

$h(0), h(1)$ both begin with a

$\Rightarrow$ $h(x)$ cannot begin with b

$\Rightarrow$ if $h^{-1}(w)$ is non-empty and $w \in L_2$, then w begins with a

$\Rightarrow$ either $w = a \longrightarrow h^{-1}(w) = \phi$

or $w = abw'$ when $w' \in L_2$

The Pumping Lemma for regular language

A powerful tool for proving certain languages nonlinear.

- L is regular if it is accepted by a DFA $M = (Q, \Sigma, \delta, q_0, F)$ with finite number of states.
- $|Q| = n$ (say)
- input of length $\geq n$, $z = a_1 a_2 \dots a_m$, $m \geq n$

Let $\delta(q_0, a_1 a_2 \dots a_i) = q_i$ for $i = 1, 2, \dots m$.

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \cdots \xrightarrow{a_i} q_i$$

$q_0, q_1, \dots q_n \rightarrow n+1$ states not distinct as M has only n states.

$\Rightarrow \exists$ integers j, k, $0 \leq j < k \leq n$ s.t. $q_j = q_k$

$$\begin{tikzcd} q_0 \arrow[r, "a_1 a_2 \dots a_j"] & q_j = q_k \arrow[loop above, "a_{j+1} \dots a_k"] \arrow[r, "a_{k+1} \dots a_m"] & q_m \end{tikzcd}$$

$z = uvw$ where $u = a_1 a_2 \dots a_j$, $v = a_{j+1} \dots a_k$, $w = a_{k+1} \dots a_m$

as $j < k$, $|v| \geq 1$, as $k \leq n$, $|uv| \leq n$

if $q_m \in F$ i.e. $a_1 a_2 \dots a_m \in L(M)$, then $a_1 a_2 \dots a_j a_{k+1} \dots a_m \in L(M)$ as

$$\begin{aligned} \hat{\delta}(q_0, a_1 \dots a_j a_{k+1} \dots a_m) &= \hat{\delta}(\hat{\delta}(q_0, a_1 \dots a_j), a_{k+1} \dots a_m) \\ &= \hat{\delta}(q_j, a_{k+1} \dots a_m) \\ &= q_m \in F \end{aligned}$$

- if $q_m \in F$ then $(a_1 \ldots a_j (a_{j+1} \ldots a_k)^i a_{k+1} \ldots a_m) \in L(M)$ $\forall i \geq 0$

i.e. given any sufficiently long string accepted by a DFA, we can find a substring near the begining of a string that may be pumped as many times as we like and the resulting string will be accepted by the DFA.

Hence follows the following lemma.

Lemma:

Let L be a regular set

- Then there is a constant n such that if z is any word in L, and $|z| \geq n$, we may write $z = uvw$ in such a way that $|uv| \leq n$, $|v| \geq 1$ and for all $i \geq 0$, uv^iw is in L
- Furthermore n is no greater than the number of states of the smallest DFA accepting L.

Note:

- if a regular set has a long string $z = uvw$ then the regular set contains an infinite number of strings of the form uv^iw

$\not\Rightarrow$ every sufficiently long string in a regular set is of the form uv^iw

Applications of the Pumping Lemma

Example: Prove that

$L = \{ 0^{i^2} \mid i \text{ is an integer}, i \geq 1 \}$ is not regular

Suppose L is regular and n be the integer in the pumping lemma.

Let $z = 0^{n^2} = uvw \in L$, $|v| \geq 1$, $|uv| \leq n$, $|v| \leq n$

Then, $uv^iw \in L \ \forall\ i \geq 0$, i.e. uv^iw should have a length which is a perfect square

<u>i = 2</u>

$$|uv^2w| = |uvw| + |v|$$

$$\geq |uvw| \text{ as } |v| \geq 1 \quad \text{and } |uvw| = n^2$$

$$\Rightarrow \quad n^2 < |uv^2w| \leq n^2 + n < (n+1)^2$$

$\Rightarrow$ length of uv^2w cannot be a perfect square

$\Rightarrow$ $uv^2w \notin L$

L is not regular.

Example: Prove that

$L = \{$Palindromes over $(0+1)^*\}$ is not regular

Let L be regular and n be the integer in the pumping lemma.

Consider $z = 0^n 1 0^n$, $|z| > n$.

By the pumping lemma, $z = uvw$, $|uv| \leq n$, $|v| \geq 1$, $|v| \leq n$

As $|uv| \leq n$, we have $u = 0^i$, $v = 0^j$

Then by Pumping Lemma, $uv^iw \in L$, $\forall\ i \geq 0$

For $i = 0$, $uv^0w = 0^{n-j} 1 0^n$

which is <u>not</u> a palindrome.

Hence a contradiction.

Example: $L = \{0^n 1^n \mid n \geq 1\}$ is not regular.

Let L be regular and n be the integer in the pumping lemma. Let $z = 0^n 1^n \in L$, $n \geq 1$

Then $|z| \geq n$ and $z = uvw$, $|uv| \leq n$, $1 \leq |v| \leq n$

and $uv^i w \in L \ \forall\ i \geq 0$.

For $i = 0$, $uv^0 w = 0^{n-j} 1^n \notin L$

$\therefore$ L cannot be regular.

Example: $L = \{1^p \mid p \text{ is prime}\}$ is not regular.

Let L be regular and n be the integer in the pumping lemma.

Consider a prime $p \geq n+2$

Let $z = uvw = 1^p \in L$, $p \geq n+2$

$\therefore$ $|z| \geq n$, $|uv| \leq n$, $1 \leq |v| \leq n$, $uv^i w \in L \ \forall\ i \geq 0$

Let $|v| = m \geq 1$, then $|uv| = |uvw| - |v| = p - m$

For $i = p-m$,

$uv^{p-m} w \in L$ by the pumping lemma.

Now, $|uv^{p-m} w| = |uw| + (p-m)|v| = p - m + (p-m)m$

$= (p-m)(m+1)$

$|uv^{p-m} w| = (p-m)(m+1)$

$m+1 \neq 1$, as $\underline{m \geq 1}$, $p - m \neq 1$ as $p \geq n+2 \geq m+2$

Thus $uv^{p-m} w$ does not have a prime length which is a contradiction.

$\therefore$ L cannot be regular.

Example:

Prove that $L = \{0^n 1^n 2^n \mid n \geq 0\}$ is not regular.

Let L be regular.

Let p be the pumping length given by the pumping lemma.

Choose $z = 0^p 1^p 2^p$

As $|z| > p$, the pumping lemma guarantees z can be splitted as $z = uvw$ with $|v| \geq 1$, $|uv| \leq p$, $|v| \leq p$

and $z = uv^i w \in L \ \forall i \geq 0$.

Note that $|uv| \leq p$

$\Rightarrow u = 0^a, \ v = 0^b, \ w = 0^{p-a-b} 1^p 2^p$

For $i = 2$

$uv^2w = 0^a 0^{2b} 0^{p-a-b} 1^p 2^p$

doesnot have equal number of 0's, 1's, and 2's and so cannot be a member of L

Hence L cannot be regular.

Example

Prove that $L = \{a^{2^n} \mid n \geq 0\}$ is not regular.

Assume that $L = \{a^{2^n} \mid n \geq 0\}$ is regular and let p be the integer in the Pumping Lemma.

Let $z = a^{2^p} \in L$

As, $|z| = 2^p > p$, Pumping lemma guarantees that

$$z = uvw, \quad |v| \geq 1, \quad |uv| \leq p \text{ and } |v| \leq p$$

and $z = uv^i w \in L \;\; \forall \; i \geq 0$

For $i = 2$

$$|uv^2w| = |uvw| + |v| = 2^p + |v| < 2^p + p$$

$$< 2^p + 2^p = 2^{p+1}$$

Also,

$$|uv^2w| = 2^p + |v| > 2^p \text{ as } |v| \geq 1$$

Thus we have

$$2^p < |uv^2w| < 2^{p+1}$$

$\Rightarrow$ $|uv^2w|$ cannot be a power of 2

$\Rightarrow$ $uv^2w \notin L$

which is a contradiction.

Hence L is not a regular

Example

Show that L is not a regular set where

L: set of all binary strings with unequal number of 0's and 1's

$= \{0, 1, 00, 11, 010, 101, \ldots\}$

Although L is not regular, we cannot prove this using Pumping Lemma (WHY?).

Let $\underline{n}$ be the integer in the Pumping Lemma.

For any word z, with $|z| \geq n$,

$z = uvw \in L$, $|uv| \leq n$, $1 \leq |v| \leq n$

$uv^iw \in L \quad \forall\, i \geq 0$

here,

v may be 01 (equal no. of 0's and 1's)

So, whenever v is pumped, number of 0's and 1's change equally

$\Rightarrow$ can never get $uv^iw \notin L$ for any i

as $uvw \in L$ $\Rightarrow$ uvw has unequal no. of 0's and 1's

$\Rightarrow$ uv^iw has unequal no. of 0's and 1's $\forall\, i \geq 0$

$\Rightarrow$ $uv^iw \in L \quad \forall\, i \geq 0$

$\Rightarrow$ L is regular.

The Pumping Lemma <u>does not</u> work for this problem

How to prove L is non-regular?

$L \rightarrow$ set of binary strings with unequal number of 0's and 1's

$\bar{L} \rightarrow$ set of binary strings with equal number of 0's and 1's

Apply the pumping lemma to $\bar{L}$

$n \rightarrow$ integer of the pumping lemma

$uvw = 0^n 1^n \in \bar{L}$ $\quad |uv| \leq n, \quad 1 \leq |v| \leq n$

$uv^i w \in \bar{L}$

for $i = 0$,

$uv^0 w \in \bar{L}$

$\Rightarrow 0^{n-|v|} 1^n \in \bar{L}$

which is a contradiction

$\therefore \bar{L}$ is not regular

$\Rightarrow L$ is not regular.

Example:

Prove that $L = \{0^m 1^n \mid m \neq n\}$ is not regular.

Observe that $\bar{L} \cap 0^* 1^* = \{0^n 1^n \mid n \geq 0\}$

If L were regular, then $\bar{L}$ would be regular and so would $\bar{L} \cap 0^* 1^* = \{0^n 1^n \mid n \geq 0\}$

But we already know by Pumping Lemma on regular languages that $\{0^n 1^n \mid n \geq 0\}$ is not regular.

$\Rightarrow$ L cannot be regular.

Alternative Method (Using Pumping Lemma directly)

Suppose L is regular and p be the integer in Pumping lemma.

Let $z = 0^p 1^{p+p!} = uvw \in L$, $|v| \geq 1$, $|uv| \leq p$, $|v| \leq p$

Observe that $|z| = p + p + p! > p$

Then by Pumping Lemma, $uv^i w \in L \;\forall\; i \geq 0$

As, $|uv| \leq p$, we have $u = 0^a$, $v = 0^b$, $w = 0^{p-a-b} 1^{p+p!}$

where $b \geq 1$ and $b \leq p$

Let $i = \frac{p!}{b}$ as $1 \leq b < p$ and $\underline{b \text{ divides } p!}$

But, $uv^{i+1}w = 0^a 0^{p!+b} 0^{p-a-b} 1^{p+p!}$

$= 0^{p!+p} 1^{p!+p} \notin L$

Hence L is not regular.

Example:

Prove that $L = \{x \in \{a,b\}^* \mid \# a(x) \neq \# b(x)\}$ is not regular

Consider $\bar{L} = \{x \in \{a,b\}^* \mid \# a(x) = \# b(x)\}$

L is non-regular if $\bar{L}$ is non regular.

Suppose $\bar{L}$ is regular, then

$$\bar{L} \cap a^* b^* = \{a^n b^n \mid n \geq 0\}$$

would also be regular, since regular sets are always closed under intersection.

But $\{a^n b^n \mid n \geq 0\}$ is a non-regular set

$\Rightarrow$ $\bar{L}$ cannot be regular

$\Rightarrow$ L is non-regular.

Example

Prove that $L = \{a^n b^m \mid n \geq m\}$ is not regular

Consider $L^R = \{b^m a^n \mid n \geq m\}$

If L is regular, then so is L^R by closure property of regular languages under reversal.

$\therefore$ interchanging a and b, we get

$$L' = \{a^m b^n \mid n \geq m\}$$

"interchanging a and b" means applying the homomorphism

$$a \rightarrow b, \quad b \rightarrow a$$

L' would also be regular if L is regular by closure property of regular languages under homomorphism

Then the intersection

$$L \cap L' = \{a^n b^n \mid n \geq 0\}$$

would also be regular

But we have already shown using Pumping Lemma that $\{a^n b^n \mid n \geq 0\}$ is not regular.

$\Rightarrow$ L <u>must be</u> non-regular language.

Direct Application of the Pumping Lemma

Let $L = \{a^n b^m \mid n \geq m\}$ be regular and $p+j+k$ be the pumping length given by the pumping lemma and $j+k < p \Rightarrow p+j+k \leq 2p$

Consider $z = a^p b^p = uvw$, $|v| \geq 1$, $|uv| \leq p+j+k$
$|v| \leq p+j+k$

As, $|z| = 2p > p+j+k$, the lemma can be applied.

$$z = a^p b^j b^k b^{p-j-k}, \quad k \geq 1$$

and $uv^i w \in L \ \forall \ i \geq 0$

For $i = 2$

$$uv^2w = a^p b^{p+k} \notin L$$

Hence L is not regular.

Example:

$L = \{a^{n!} \mid n \geq 0\}$ is not regular

Let L be regular and K be the pumping length given in the pumping lemma

Consider $z = a^{K!}$

Then $|z| = K! > K$

Hence by the pumping lemma on regular languages

$z = uvw$ with $|v| \geq 1$, $|uv| \leq K$, $|v| \leq K$

and $uv^iw \in L$ $\forall i \geq 0$.

Note that

$$|uvw| = K!$$

$$|uv^iw| = K! + i|v|$$

for $i = (K+1)!$

$$|uv^iw| = K! + ((K+1)!)|v|$$

$$= K!(1 + (K+1)|v|)$$

$$\neq p! \text{ for any } p \quad (p > K)$$

as for $p > K$, $p!$ is divisible by $(K+1)$ but $K!(1+(K+1)|v|)$ is not.

Theorem:

The set of strings accepted by a FA M with n states is:

(i) non-empty iff the finite automata accepts a string of length $<n$

(ii) infinite iff M accepts some strings of length l, when $n \leq l < 2n$

Proof:

(i) if M accepts a string of length $<n$, then M is non-empty

Conversely,

if M is non-empty, let $w \in L(M)$ be a string as short as any other string accepted.

Then by the Pumping lemma, $|w|<n$, otherwise if $|w|\geq n$, then $w = uvy$, $1 \leq |v| \leq n$ and $uy \in L(M)$ where uy has shorter length than w, a contradiction

(ii) if M accepts a string w, when $n \leq |w| < 2n$, then by Pumping lemma, $L(M)$ is infinite

$(|w| \geq n \Rightarrow w = uvy,\ 1 \leq v < n \text{ and } uv^iy \in L,\ \forall i \geq 0)$

Conversely,

if $L(M)$ is infinite, then $\exists\ w \in L(M)$, where $|w| \geq n$

Now, if $\underline{|w| < 2n}$, we are done

if no word is of length between $\underline{n \text{ and } 2n-1}$, let w be of length atleast $2n$, but as short as any word in $L(M)$ whose length is $\geq 2n$

Again by Pumping Lemma, we can write

$w = uvy$ with $1 \leq |v| \leq n$ and $uy \in L(M)$

uy has shorter length than w

$\therefore$ $|uy| \not\geq 2n$ as w is the shortest string with length $\geq 2n$

$\Rightarrow$ $n < |uy| < 2n-1$, contradicting no word in $L(M)$ with length between n and $2n-1$

Hence, the result.

Emptiness Check

Check if any word of length upto n is in $L(M)$

$\Rightarrow$ exponential time

(exhaustive search 2^n over $\{0,1\}^*$)

Infinite / finiteness chech

Check if any word of length between n and $2n-1$

$\rightarrow$ $O(2^n)$.

Decision algorithms for regular languages.

- Algorithm to decide whether two DFA are equivalent or not

$M_1 \xrightarrow{DFA}$ accepts L_1

$M_2 \xrightarrow{DFA}$ accepts L_2

$(L_1 \cap \bar{L}_2) \cup (\bar{L}_1 \cap L_2) \rightarrow$ regular language by closure property

$\rightarrow$ accepted by DFA M_3

M_3 accepts a string iff $L_1 \neq L_2$

- Algorithm to decide whether a regular set is empty, finite or infinite

Theorem:

The set of strings accepted by a finite automata M with n states is

(i) nonempty iff M accepts a strings of length $\leq n$

(ii) infinite iff M accepts a string of length l, where $n \leq l < 2n$

Theorem

The class of regular sets is closed under homomorphisms and inverse homomorphisms.

Proof:

Closure under homomorphism: follows from closure under substitution as every homomorphism is a substitution, in which $h(a)$ has one member.

Closure under inverse homomorphism

- $M = (Q, \Sigma, \delta, q_0, F)$: DFA accepting L
- $h \rightarrow$ homomorphism from $\Delta \rightarrow \Sigma^*$

Construct a DFA,

$M' = (Q, \Delta, \delta', q_0, F)$ that accepts

$h^{-1}(L) = \{x \mid h(x) = w \in L\}$

and $\delta'(q, a) = \hat{\delta}(q, h(a))$

$h(a)$ can be ε or a long string

Claim:

$\delta'(q_0, x) = \hat{\delta}(q_0, h(x))$, i.e. M' accepts x iff M accepts $h(x)$

i.e. $L(M') = h^{-1}(L(M))$

This can be proved by induction on $|x|$

Better algorithm for checking emptiness of regular languages

- Given a DFA, if the accepting states are all separated from the start state, then the language is empty.

Graph reachability : Decide whether we can reach an accepting state from the start state.

$O(n^2)$ if FA has n states.

Construct the set of reachable states from q_0, if this set contains an accepting state, output NO (i.e. language of the FA is not empty) otherwise output YES

- Given a regular expression of language L, convert the regular expression to ε-NFA and apply the above algorithm.

  $O(n^2)$ if the regular expression has length n.

- Given a regular expression R of language $L(R)$, to check whether $L(R)$ is empty or not ?

Case I: $R = R_1 + R_2 \Rightarrow L(R)$ is empty iff $L(R_1)$, $L(R_2)$ both are empty

Case II: $R = R_1 R_2 \Rightarrow L(R)$ is empty iff either $L(R_1)$ or $L(R_2)$ is empty.

Case III: $R = R^* \Rightarrow L(R)$ non-empty, contains ε

Case IV: $R = R_1 \Rightarrow L(R)$ is empty iff $L(R_1)$ is empty

Finiteness/Infiniteness Checking

- DFA accepts an infinite language iff the resulting transition diagram has a cycle
- Same method works for NFA's, but we must check that there is a cycle labeled by something besides ε

Identifiers for regular expressions (r.e.).

1. $\phi + r = r$, r is r.e.

2. $\phi r = r\phi = \phi$

3. $\varepsilon r = r\varepsilon = r$

4. $\varepsilon^* = \varepsilon$ and $\phi^* = \varepsilon$

5. $r + r = r$

6. $r^* r^* = r$

7. $r r^* = r^* r$

8. $(r^*)^* = r^*$

9. $\varepsilon + r r^* = r^* = \varepsilon + r^* r$

10. $(r s)^* r = r (s r)^*$

11. $(r + s)^* = (r^* s^*)^* = (r^* + s^*)^*$

12. $(r + s) t = rt + st$, $t (r + s) = tr + ts$

Arden's Theorem

Let $\underline{r}$ and $\underline{s}$ be two regular expressions over Σ

If r does not contain ε, then the following equation in X namely

$$X = s + Xr \quad \text{——— (1)}$$

has a unique solution given by $X = sr^*$

Proof:

Existence: $s + (sr^*)r = s(\varepsilon + r^*r) = sr^*$

$\Rightarrow X = sr^*$ is a solution of (1)

Uniqueness:

Replacing X by $s + Xr$ on R.H.S. of (1) yields

$$\begin{aligned} X = s + Xr &= s + (s + Xr)r \\ &= s + sr + Xr^2 \\ &= s + sr + sr^2 + Xr^3 \\ &= s + sr + \dots + sr^i + Xr^{i+1} \end{aligned}$$

Thus we have, $X = s(\varepsilon + r + r^2 + \dots + r^i) + Xr^{i+1}$, $i \geq 0$

(2)

Claim: Any solution of (1) is equivalent to sr^*

Note that

X satisfies (1)

$\Rightarrow X$ satisfies (2)

Let $w \in X$ with $|w| = i$

Then $w \in s(\varepsilon + r + \dots + r^i) + Xr^{i+1}$

As r doesnot contain ε, Xr^{i+1} has no string of length $< i+1 \Rightarrow w \notin Xr^{i+1}$

$\Rightarrow w \in s(\varepsilon + r + r^2 + \dots + r^i)$

$\Rightarrow w \in sr^*$

$\Rightarrow X \subseteq sr^*$ ——— (3)

Now consider any $w \in sr^*$

Then $w \in sr^k$ for some $k \geq 0$

$\Rightarrow w \in s(\varepsilon + r + r^2 + \dots + r^k)$

$\Rightarrow w \in$ R.H.S. of (2)

$\Rightarrow w \in X \Rightarrow w$ is a solution of (1)

$\Rightarrow sr^* \subseteq X$ ——— (4)

From (3) and (4), we have

$X = sr^*$

Applications of Arden's Theorem
(Brzozowski's Algebraic Method).

The following assumptions are made regarding the transition system:

i. The transition graph doesnot have ε- moves

ii. It has only one initial state, say v_1

iii. It's vertices are $v_1, v_2, \ldots, v_n$

iv. v_i the regular expression represents the set of strings accepted by the system even though v_i is an accepting state.

v. α_{ij} the r.e. representing the set of labels of edges from v_i to v_j
($\alpha_{ij} = \phi$ if no such edge)

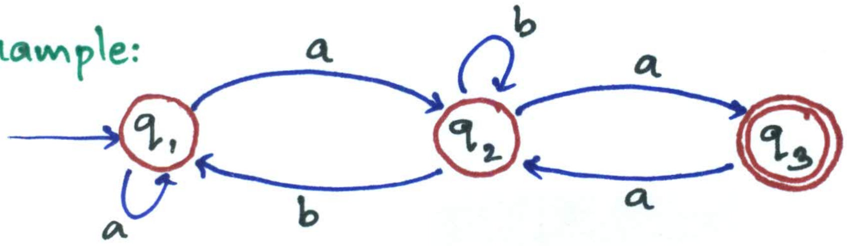
Construct the following set of equations in $v_1, v_2, \ldots v_n$

$$v_1 = v_1\alpha_{11} + v_2\alpha_{21} + \cdots + v_n\alpha_{n1} + \varepsilon$$
$$v_2 = v_1\alpha_{12} + v_2\alpha_{22} + \cdots + v_n\alpha_{n2}$$
$$\vdots$$
$$v_n = v_1\alpha_{1n} + v_2\alpha_{2n} + \cdots + v_n\alpha_{nn}$$

Repeatedly apply substitution and Arden's theorem to express v_i in terms of α_{ij}'s

Union of all v_i corresponding to final states
$\rightarrow$ the set of strings recognized by the transition system.

Example:



- no ε - moves
- only 1 initial state q_0

Construct three equations corresponding to three vertices q_1, q_2, q_3

$$Q_1 = Q_1 a + Q_2 b + \varepsilon \quad \text{———} (1)$$

$$Q_2 = Q_1 a + Q_2 b + Q_3 a \quad \text{———} (2)$$

$$Q_3 = Q_2 a \quad \text{———} (3)$$

(2) and (3) gives

$$Q_2 = Q_1 a + Q_2 b + Q_2 aa$$

$$= Q_1 a + Q_2 (b+aa) = Q_1 a (b+aa)^* \quad \text{———} (4)$$

↳ using Arden's theorem

(1) and (4) gives:

$$Q_1 = Q_1 a + Q_1 a (b+aa)^* b + \varepsilon$$

$$= \varepsilon + Q_1 (a + a(b+aa)^* b)$$

$$= \varepsilon (a + a(b+aa)^* b)^*$$ - by Arden's Theorem

Thus,

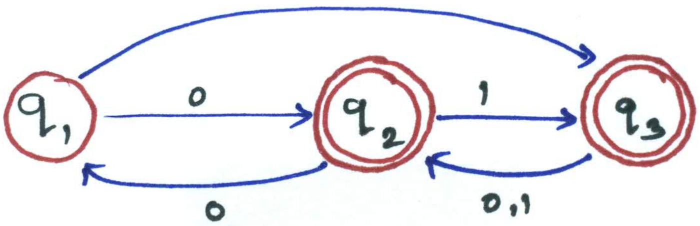
$$Q_1 = (a + a(b+aa)^* b)^*$$

$$Q_2 = (a + a(b+aa)^* b)^* a (b+aa)^*$$

$$Q_3 = (a + a(b+aa)^* b)^* a (b+aa)^* a$$

Since q_3 is the final state, the set of strings recognized by this graph is given by regular expression Q_3

Example:



r.e. $Q_2 + Q_3 = ?$

$$Q_1 = Q_1\phi + Q_2 0 + Q_3\phi + \varepsilon = Q_2 0 + \varepsilon$$

$$Q_2 = Q_1 0 + Q_2\phi + Q_3(0+1) = Q_1 0 + Q_3(0+1)$$

$$Q_3 = Q_1 1 + Q_2 1 + Q_3 0 = Q_1 1 + Q_2 1$$

$$\Rightarrow \quad Q_2 = Q_1 0 + (Q_1 1 + Q_2 1)(0+1)$$

$$= Q_1 0 + Q_1 1(0+1) + Q_2 1(0+1)$$

$$= Q_1(0+1(0+1)) + Q_2 1(0+1)$$

$$= Q_1(0+1(0+1))(1(0+1))^*$$

$$= (Q_2 0+\varepsilon)(0+1(0+1))(1(0+1))^*$$

$$= X + Q_2 0X$$

$$= X(0X)^*$$

$$= (0+1(0+1))(1+(0+1))^*\left(0(0+1(0+1))(1(0+1))^*\right)^*$$

$$Q_3 = Q_1 1 + Q_2 1$$

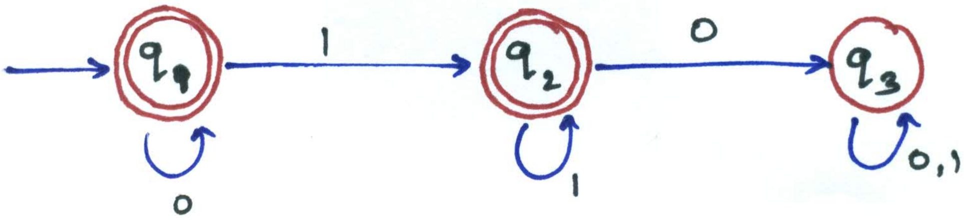
$$= (Q_2 0+\varepsilon)1 + Q_2 1$$

$$= 1 + Q_2(01+1)$$

$$= 1 + X(0X)^*(01+1)$$

where $X = (0+1(0+1))(1(0+1))^*$

Example: Construct a r.e. corresponding to the state diagram described by



$$q_1 = q_1 0 + \varepsilon = \varepsilon + q_1 0$$

$$q_2 = q_1 1 + q_2 1$$

$$q_3 = q_2 0 + q_3 (0+1)$$

$\Rightarrow$

$$q_1 = \varepsilon + q_1 0$$
$$= \varepsilon 0^* = 0^* \quad \text{(Arden's theorem)}$$

$$q_2 = q_1 1 + q_2 1$$
$$= 0^* 1 + q_2 1$$
$$= (0^* 1) 1^*$$

No need to solve for q_3 as final states are q_1 and q_2

$\therefore$

$$q_1 + q_2 = 0^* + (0^* 1) 1^*$$
$$= 0^* + 0^* (1 1^*)$$
$$= 0^* (\varepsilon + 1 1^*)$$
$$= \underline{\underline{0^* 1^*}}$$