Set Theory Basics

- Types of definition
  - By listing
    - Eg $S = \{1, 2, 3, \ldots\}$
  - By description
    - Eg $S = \{x \mid x \in \mathbb{N} \text{ and } x > 8\}$
  - By induction
    - Eg. Basis: $1 \in \mathbb{N}$
      Inductive step: if $x \in \mathbb{N}$, then $x + 3 \in \mathbb{N}$.

- The Power Set
  - A set of all subsets of the given set.
    - Eg. $P(\mathbb{N}) = \{x \mid x \subseteq \mathbb{N}\}$
    - Eg. $S = \{0, 1\}$
      $P(S) = \{\{0\}, \{1\}, \{0, 1\}, \emptyset\}$

- Basic Symbols
  - $\cup$ Union
  - $\cap$ Intersection
  - $\subset$ Proper subset
  - $\subseteq$ Subset
  - $\supset$ Proper superset
  - $\supseteq$ Superset

"Proper" means not equal, so $\{0,1\} \subseteq \{0,1\}$ but $\{0,1\} \not\subset \{0,1\}$

Sequences

- A list of things in order
  - Normal brackets
  - Order and repetition do matter

- A finite sequence is a tuple. A k-tuple is a tuple with k elements. A 2-tuple is an ordered pair.

- Sequences are equal if the values in every position are equal
  Eg. $(0,1,2) = (0,1,2)$ but $(0,1,2) \neq (0,2,1)$

- **Cartesian Product**

$$A \times B = \{(a,b) \mid a \in A \text{ and } b \in B\}$$

$$A_1 \times \cdots \times A_n = \{(x_1, \cdots, x_n) \mid x_i \in A_i \text{ for } i = 1 \to n\}$$

- **Binary Relations**
  - The B.R. of sets A and B is any subset (R) of the Cartesian product $A \times B$
    - aRb denotes $(a,b) \in R$
    - $a \not R b$ denotes $(a,b) \notin R$

- **Properties of Relations**

  - Relation R on set A is reflexive if

$$(a,a) \in R \text{ for every } a \in A$$

    - ie. every element loops

- Relation R on set A is **irreflexive** if

$(a,a) \notin A$ for every $a \in A$

° ie. no element loops 

- Relation R on set A is **symmetric** if

$(a,b) \in R$ wherever $(b,a) \in R$, for every $a,b \in A$

° ie. all relations are 2-way 

- Relation R on set A is **anti-symmetric** if

(a,b) and (b,a) cannot both be in R, unless $a=b$.

° ie. no two-way relations 

- Relation R on set A is **transitive** if the following holds:

if (a,b) and $(b,c) \in R$, then $(a,c) \in R$ for all $a,b,c \in R$

° ie. anything that can be done in 2 or more steps can also be done in 1 single step

• Transitive Closure

If R is a relation on set A, the T.C. of R is the smallest transitive relation on A that contains R.

R^* is the T.C. of R
If R is transitive, $R = R^*$

- Given R, R^* can be defined inductively:

Basis: $R \subseteq R^*$

Inductive: If $(a,b) \in R^*$ and $(b,c) \in R^*$
$(a,c) \in R^*$

• Computing The Transitive Closure

- Warshall's algorithm computes R^* given the matrix of R
  - R is the rel on a set w/ n elements
  - Start with $M_0 = R$
  - There are n rounds. Let k be the counter from 1 to n.
  - For each k:
    - Study each element of M_{k-1}
    - M_{ij} = the element in row i, col j
      - If $M_{ij} = 1$, skip
      - If $M_{ik} = M_{kj} = 1$, change M_{ij} to 1
    - Call the resulting matrix M_k
  - $M_n = R^*$

• Partial Orders / Equivalence Relations

- A relation is a partial order if it is...
  - Reflexive
  - Antisymmetric
  - Transitive

- A relation is an equivalence if it is...
  - Reflexive
  - Symmetric
  - Transitive

STEP ONE

$$M_0 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \rightarrow M_1 = \begin{bmatrix} & 1 & 1 & 1 \\ & 1 & 1 & 1 \\ 1 & & & \\ & & 1 & 1 \end{bmatrix}$$

STEP TWO

$$M_1 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & & & \\ 0 & & 1 & 1 \end{bmatrix}$$

STEP THREE

$$M_1 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ & & ? & \\ 0 & & 1 & 1 \end{bmatrix}$$

STEP FOUR

$$M_1 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \rightarrow M_2 = \begin{bmatrix} & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ & 0 & 1 & 1 \end{bmatrix}$$

• Hasse Diagram

- Used when a relation is known to be a partial order.

- From a directed graph...
  - Remove loops
  - Remove arrows that are only there b/c of transitivity
  - Rearrange so that arrows point "up"
  - Remove arrow heads.

• Linear Order

- A relation is a linear / total order if...
  - R is a partial order
  - Every given pair of elements is connected one way or another

• Composition

- If R is a relation from sets A to B
  S is a relation from sets B to C

$$R \circ S = \{(a,c) \in A \times C \mid (a,b) \in R \text{ and } (b,c) \in S \text{ for some } b \in B\}$$

- N.B. $R \circ S \neq S \circ R$

- Can be seen as connecting two relations where a "through" path exists

Functions

- A func. from set A to set B maps every element of set A to exactly one element of set B.

- Rules:
  - Every element of A must map to B
  - Not every element in B must be mapped from A
  - An element in A must map to exactly one element in B
  - Two+ elements in A can map to the same element in B

• Multiple Arguments

- If the domain of f is a Cartesian product, we say f has arity of n, f is n-ary or f has n arguments

• Tuples / Sequences as Functions

- The 4-tuple $(4, 12, 15, 11)$ can be thought of as listing the values of $f: \{4, 12, 15, 11\} \to \mathbb{N}$ defined by $f(0)=4$, $f(1)=12$, etc.

• Functions as Binary Relations

- A function $f: A \to B$ can be seen as a relation from A to B:

$$R = \{(a, b) \in A \times B \mid f(a) = b\}$$

• Properties

- **Injective** if it is one-to-one ($x \neq y$ implies $f(x) \neq f(y)$)
- **Surjective** if every element of the codomain is mapped to
- **Bijective** if both of the above.

• Bijections and Inverses

- Bijections **work in pairs**:

If $f(a) = b$, $f^{-1}(b) = a$

If $f: A \to B$, $f^{-1}: B \to A$

• Useful Functions

- **Identity Function**
  - $id_A: A \to A$, such that $id_A(a) = a$ for all $a \in A$.
  - id_A is a bijection, and the inverse is itself.

- **Characteristic Set.**
  - For every $a \in A$ where $A \subseteq S$ and S is any set.
  - $f_A: S \to \{0,1\}$

$$f_A(x) = \begin{cases} 0 & \text{if } x \notin A \\ 1 & \text{if } x \in A \end{cases}$$

• Describing $\mathbb{N} \to \mathbb{N}$ by Induction

- Fibonacci Function:
  - Basis: $f(0) = 1$ and $f(1) = 1$
  - Inductive: if $n > 1$, $f(n) = f(n-1) + f(n-2)$

• Combining functions: Compositions

- Let $g: A \to B$ and $f: B \to C$

$$(f \circ g)(x) = f(g(x)) \quad \leftarrow g() \text{ is done first.}$$

- $f \circ g$ is only defined if the domain of f is the codomain of g
  - Eg. $g: A \to B$ $f: B \to C$ $\therefore f \circ g$ is okay
    $g: A \to B$ $f: A \to C$ $\therefore f \circ g$ is undefined.

- $f \circ g$ and $g \circ f$ can be different

- $f \circ (g \circ h) = (f \circ g) \circ h$ ($\circ$ is associative)

- if $f: A \to B$ is a bijection, then
  $f^{-1} \circ f = id_A$ and $f \circ f^{-1} = id_B$

- for any $f: A \to B$, $f \circ id_A = id_B \circ f = f$

• Comparing Finite Sets

- Two finite sets with the same length always have a bijection between them.

- A set is countable if there is a bijection between it and $\mathbb{N}$.

Sets - Counting and Selection

- $|S|$ is the length of the finite set S.
  - $|\emptyset| = 0$

• The Sum Rule

- If A and B are disjoint sets, $|A \cup B| = |A| + |B|$

• Inclusion / Exclusion Principle

- $|A \cup B| = |A| + |B| - |A \cap B|$

- $|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |A \cap C| + |A \cap B \cap C|$

• The Product Rule

- $|A_1 \times \cdots \times A_n| = |A_1| \times \cdots \times |A_n|$

  ↑ Cartesian Product ↑ Multiplication

- If $|A| = n$ and $|B| = m$, the number of $f: A \to B$ functions is

$$\underbrace{m \times m \times \ldots}_{n} = m^n$$

• Counting Subsets

- The number of possible subsets in a n-length set is 2^n
- $|P(S)| = 2^n$ if $|S| = n$

• Pigeonhole Principle

- If n objects are to be placed in k boxes, at least one box contains at least $\lceil n/k \rceil$ objects

• Selection Methods

- A bag has 3 items; A, B and C. How many ways to pick 2?
  - Order matters, No repetition
    → AB, AC, BA, BC, CB, CA
  - Order matters, Repetition okay.
    → AA, AB, AC, BA, BB, BC, CA, CB, CC
  - Order irrelevant, No repetition
    → AB, BC, AC
  - Order irrelevant, Repetition Okay
    → AA, AB, AC, BB, BC, CC

• Order Matters / No Repetition

"How many k-length photo lineups from an n-number group?"

- n options for the first number
- (n-1) options for the second number
- And so on, up to n-(k-1) for the k^{th} number

$$n \times (n-1) \times \cdots \times n-(k-1) = \frac{n!}{(n-k)!}$$

• Order Matters / Repetition Okay

"How many k-letter words from an n-letter alphabet?"

- n ways for the first letter
- n ways for the second letter
- And so on, up to n ways for the k^{th} letter

$$\underbrace{n \times n \times \cdots \times n}_{k} = n^k$$

• Order Irrelevant / No Repetition

"How many k-people groups from a n-size class?"

- With order important, we use $\frac{n!}{(n-k)!}$
- But each set of k people was counted k! times, so divide by k!

$$\frac{n!}{(n-k)!} \div k! = \frac{n!}{k! \times (n-k)!} = \binom{n}{k} = \frac{n \times (n-1) \times \cdots \times n-(k-1)}{1 \times 2 \times \cdots \times k}$$

"n choose k"

• Order Irrelevant / Repetition Okay.

"How many k-component mixes can be make from an n-colour set of options, where colours can be picked more than once?"

$$\binom{k+n-1}{n-1} = \binom{k+n-1}{k}$$

• Summary of Counting Selections:

- When picking k items from n options...

	Order Matters (permutations)	Order Irrelevant (combinations)
No Repetition	$\frac{n!}{(n-k)!}$	$\binom{n}{k}$
Repetition Okay	n^k	$\binom{k+n-1}{k}$

• Binomial Theorem

- Let $n, k \in \mathbb{N}$, $n \geq k$. Then $\binom{n}{k}$ is a binomial coefficient
- This is b/c these numbers appear as coefficients in the expansion of powers of binomial expressions
- Eg. $(a+b)^n = \binom{n}{0}a^n b^0 + \binom{n}{1}a^{n-1}b^1 + \cdots \binom{n}{n-1}a^1 b^n$
- Properties:
  - $\binom{n}{k} = \binom{n}{n-k}$
  - Pascal's identity: $\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k+1}$

Probability Theory

• Probability of a Union of an Event

- $P(E_1 \cup E_2) = P(E_1) + P(E_2) - P(E_1 \cap E_2)$

• Conditional Probability

- $P(A|B)$ means "prob. of A given B has happened"
- $P(A|B) = \frac{P(A \cap B)}{P(B)}$
- If $P(A|B) \neq P(A)$ then A and B are not independent

• Independence

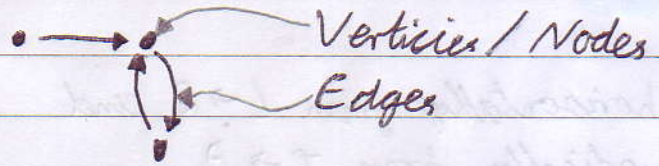
- Independent events do not impact each other
- $P(A|B) = P(A)$
- $P(A \cap B) = P(A) \cdot P(B)$

} If A and B are independent

• Bayes Theorem

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B|A) \cdot P(A) + P(B|\bar{A}) \cdot P(\bar{A})}$$

Graphs



• Terms

- Simple graph: undirected edges, no multiple edges, no loops
- Multigraphs: undirected edges, multiple edges okay, loops okay
- Directed graphs: directed edges, no multiple edges, no loops

- Directed multigraphs can show a binary relation

- If there is an edge e between nodes a and b, we say a and b are adjacent or e is incident with a and b.

- The degree of a node is the num. of connected edges it has.
  - If degree = 0, the node is isolated
  - If degree = 1, the node is pendant

- Number of edges = $\frac{\text{sum of degrees}}{2}$

- For the directed graph 

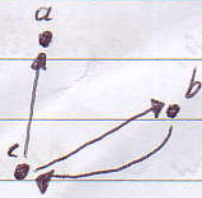
  - a is adjacent to b
  - a is the initial / source node
  - b is the terminal / end node
  - "in-degree" / "out-degree" are self-explanatory

- Number of edges = sum of in-degrees
  = sum of out-degrees

• Adjacency Matrix

- List vertices horizontally from L → R
- List vertices vertically from T → B
- The number in the i^{th} row / j^{th} column is the number of edges going **from** i **to** j

- For an undirected graph $(i, j) = (j, i)$

- Eg. 

$$= \begin{array}{c} \\ a \\ b \\ c \end{array} \begin{array}{c} \begin{array}{ccc} a & b & c \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \end{array}$$

• Paths in Simple Graphs

- A path is a sequence of nodes travelling on edges
- The length is the number of edges
- Simple path: no same edge twice
- Hamiltonian Path: simple path passing through every node once

• Cycles in Simple Graphs

- A cycle is a path that starts and ends at the same node
- Length, simplicity and Hamiltonian-ness have the same definitions as above

• Special Graphs

- The complete / n-clique / K_n graph has one edge for each node pair

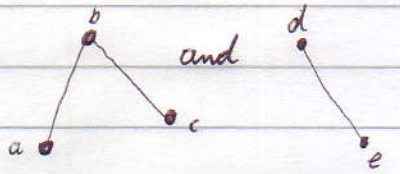
- The n-cycle / C_n graph looks like a cycle. Obviously.

- Eg: C_3 =  C_4 =  K_3 =  K_4 = 

• Sub-Graphs

- A sub-graph is an original graph with nodes or edges removed.

• Connectivity

- A graph is connected if a path exists between any two nodes
- A connected component is a maximal connected subgraph
- Eg. 

could be connected components of a larger graph.

• Isomorphism

- Graphs are isomorphic if they represent the same information
- You can change one to the other by dragging nodes but not changing connections

- Graphs are isomorphic if there is an isomorphism between them: a function f mapping the nodes of one to the nodes of the other.
  - f is a bijection
  - f maps edges to edges and non-edges to non-edges.

- An invariant is a property preserved under isomorphisms
  - num. of nodes
  - num. of edges
  - num. of x-degree nodes
  - C_n/K_n subgraphs

- To confirm an iso., find the bijective function
- To refute an iso., find an invariant that is "broken"

Trees

- A tree is a connected simple path with no cycles

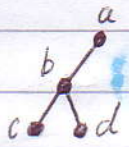
- In a tree there is one unique path between two nodes
  - Adding any edge creates a cycle
  - Removing any edge makes it non-connected

• Rooted Trees

- One node is nominated as the root, usually drawn at the top
- Any node can be a root.

- Two rooted trees are isomorphic if the bijective function maps roots to roots, edges to edges and non-edges to non-edges.

- In this rooted tree...



  - a is the parent of b
  - b is the child of a
  - c and d are siblings
  - b and a are ancestors of c
  - b, c and d are descendants of a

- The depth/level of a node is the length of the path back to root
- The height of a rooted tree is the max length from root to leaf
- A subtree is created by picking a new node to use as the root and discarding everything above it.

- A child-less node is a leaf
- A node with children is internal.

- For any $m \in \mathbb{N}^+$, a m-ary rooted tree has no internal node with more than m children.
  - A full m-ary rooted tree exists where every internal node has exactly m children

• Tree Properties

- A full m-ary tree with n internal nodes has $mn+1$ nodes overall.
  - Because each internal node has m children, plus 1 for the root

- A tree with n nodes has $n-1$ edges
  - Proof by induction: Let $n=1$, edges $= n-1$
    Then let $n=n+1$, edges $= n-1+1 = n$

• Binary Search Trees

- Given a linear order α and a list L, in the binary search tree the children of any given node are on the left if they are smaller or the right if they are larger than the given node.

- Building a binary search tree, given list L and order α:
  - Go through list L→R
  - First item: root
  - Compare: take each item and compare to existing tree, starting at the root and...
    - moving left if α-less
    - moving right if α-greater

• Tree Traversal

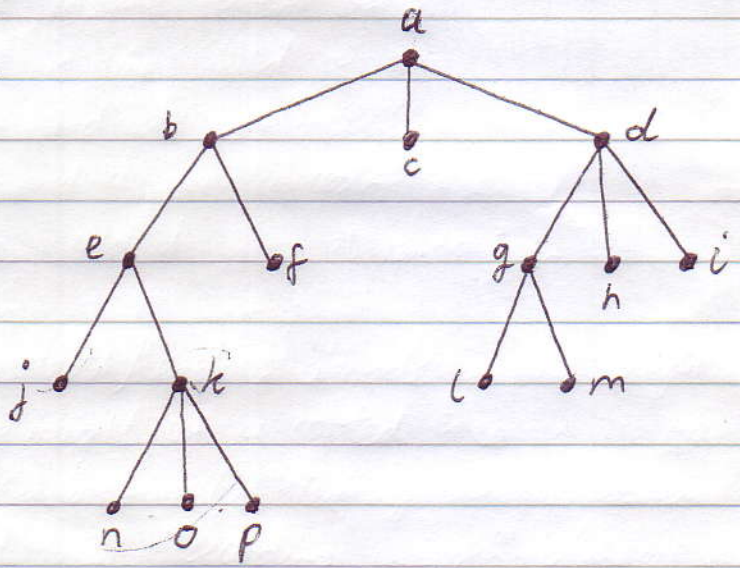
- Used to visit each node to read and process data.

- **Preorder Traversal**
  - Visit root first
  - Repeat traversal in preorder, left to right
  - Recurse until complete

- **Inorder Traversal**
  - Traverse leftmost subtree in inorder
  - Visit root
  - Traverse subtrees in inorder, left to right

- **Postorder Traversal**
  - Traverse leftmost subtree in postorder
  - Continue traversing in postorder, left to right
  - finally visit root.

- Eg.



Preorder : a b e j k n o p f c d g l m h i

Inorder : j e n k o p b f a c l g m d h i

Postorder : j n o p k e f b c l m g h i d a

Finite Automata

- aka. Finite State Machines

• Terms

- **Alphabet**: all available symbols to make a word (Σ)
- **Word**: 1 or more symbols from the alphabet.
- ϵ: the empty word
- Σ^*: set of all words from alphabet Σ (includes ϵ)
  - Σ^* is infinite if Σ is non-empty

- **Length** of word $w = |w|$
  - $|hello| = 5 \quad |\epsilon| = 0$

- xy = **concatenation** of x and y
- $x^n = xxx\ldots$
- $\epsilon x = x\epsilon = x$ for all words x
- $x^0 = \epsilon$

- A **language** is a set of words over a given alphabet
  - Can be defined as $L_1 = \{a^x b^y \mid x > y\}$
  - Can be infinite

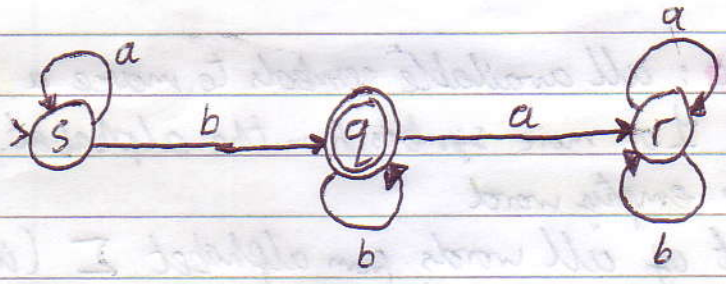
• Theoretical Model

- An F.A. has one or more states. At any time it can be in any of its states
  - Special states: one initial state, and possibly some favourable
- The F.A. is hardwired on how to move from one state to the next, usually depending on an input.
- At the end of the input, the word is **accepted / rejected** depending on if the F.A. ends in a favourable state or not.

• State Transition Diagram

- (a) = initial state
- ((b)) = favourable state
- $\xrightarrow{c}$ = move on given input

- Eg.



- Example Computation:
  - Input: abba
  - Computation: (s, abba), (s, bba), (q, ba), (q, a), (r, ε)
  - Result: rejected

Each called a configuration. (current state, remaining input)

• Deterministic Finite Automata (DFAs)

- A DFA is a 5-tuple $A = (Q, \Sigma, \delta, s, F)$ where...
  - Q is the set of states
  - Σ is the finite input alphabet
  - $s \in Q$ is the initial state
  - $F \subseteq Q$ is the set of favourable states
  - $\delta : Q \times \Sigma \to Q$ is the transition function:

If the system is in state $q \in Q$ and reads input $a \in \Sigma$, it will move to the new state $q' = \delta(q, a)$

- These are described as deterministic b/c at every state there is one unique arrow for each symbol in the alphabet.
  - ie. the pair (current state, symbol read) uniquely determine the next state.

- δ may be shown as a transition table:

$Q = \{s, q, r\}$ $\Sigma = \{a, b\}$

δ	a	b
s	s	q
q	r	q
r	r	r

- The language of a DFA is all words accepted by that DFA

• Non-deterministic Finite Automata (NFAs)

- A finite automaton is non-deterministic if, from any state, the input read does not determine exactly the next state.

- Easier to design than DFAs
  - Any NFA can be expressed as a DFA

- An NFA can have multiple computations of a word
  - A word is accepted if any computation end favourably.
  - A word being stuck in a favourable state does not count as acceptance

- An NFA is a 5-tuple $A = (Q, \Sigma, \Delta, s, F)$ where...
  - Q, Σ, s and F are the same as for DFAs
  - $\Delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Q$ is the transition relation:

    If A is in state q and reads input a, it enters any of the states q' for which $(q, a, q') \in \Delta$.

- An NFA can be spotted by:
  - Multiple same-label arrows from one state
  - Trap states
  - ϵ-jumps

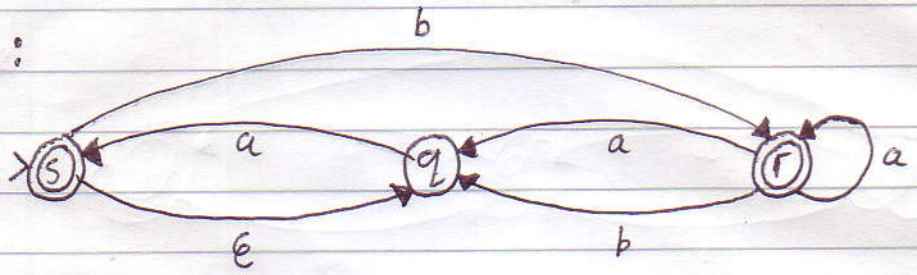
• NFA → DFA

Given... NFA = A, DFA = A'

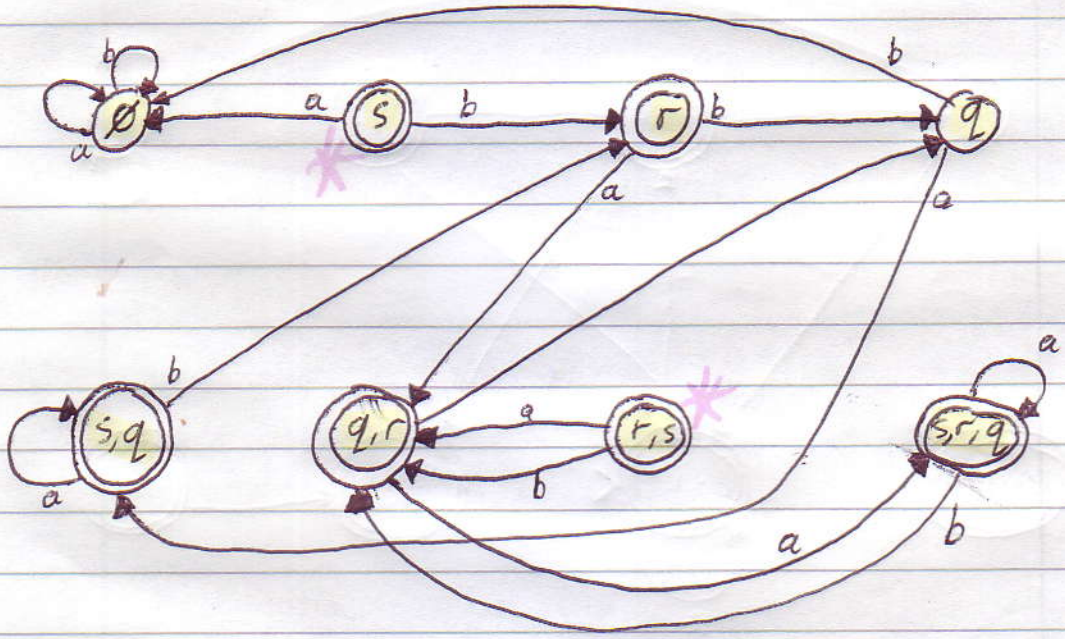
- States of A' = all subsets of A's set of states
- Initial state of A' = the set containing the initial state from A and all states reachable by an ϵ-jump from the initial state
- Favourable states of A' = all states containing at least 1 favourable state from A
- Transition formula = for every new state in A' and all $a \in \Sigma$:
  $\delta(q, a)$ = the set of states reachable from any state in q by following an a-arrow, followed by any ϵ-arrows
- Remove unreachable "extra" states

- Example:

A =



A' =



* Nothing goes to s or r,s so these can be removed.

Languages and Regular Expressions

- Many languages can be represented as a regular expressions
- $L = L[a^*b^*]$ means "L is represented by the regex a^*b^*"

• Description by Induction

- A regex over an alphabet Σ is a string consisting of...
  - symbols from Σ
  - plus symbols from * ϵ $\cup$ ( ) $\emptyset$
    (*: 0 or more; $\cup$: "or")

- The set of regexes over Σ can be described inductively:
  - Basis: $\emptyset$, ϵ and each symbol in Σ are regexes
  - Inductive: if a and b are regexes, so are $(a \cup b)$, (ab) and (a^*).
  - Closure: no other strings are regexes over Σ

• Conventions

- Brackets are omitted in concatenation, ie. $(ab)b = abb$
- * binds tighter than concatenation, so $aba^* = (ab)(a^*)$

• Representing Languages

- Every regex over Σ represents a language over Σ
  - There are two different ideas:
    - A regex = a
    - The language represented by $a = L[a]$

- $L[\emptyset] = \emptyset$ $L[\epsilon] = \{\epsilon\}$ $L[a] = \{a\}$ for any $a \in \Sigma$

- If a and b are regexes...
  - $L[a \cup b]$ = all words in $L[a]$ or $L[b]$
  - $L[ab]$ = any word in $L[a]$ followed by any word in $L[b]$
  - $L[a^*]$ = zero or more consecutive words from $L[a]$

• Regular Languages

- Can be described by a regex.
  ie. L is regular if $L = L[a]$ for some regex a.

- Not all languages are regular

- There are general procedures to convert from a regex to a NFA and from a NFA to a regex.
  - Every regular language is accepted by some automaton
  - Every language accepted by an automaton is regular

• Regex → NFA

- An inductive process using regex a to make NFA A such that $L(A) = L[a]$

- Base cases:

  - If $a = \emptyset$, then $L[a] = \emptyset$ A: 

  - If $a = \epsilon$, then $L[a] = \{\epsilon\}$ A: 

  - If $a = x$, then $L[a] = \{x\}$ A: 

- Inductive Cases:

- Automaton accepting $L[a \cup b]$

A_a: accepts $L[a]$

A_b: accepts $L[b]$

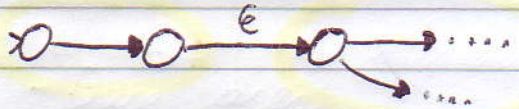
A: accepts $L[a \cup b]$

- Automaton accepting $L[ab]$

A_a: accepts $L[a]$

A_b: accepts $L[b]$

A: accepts $L[ab]$



- Automaton accepting $L[a^*]$

A_a: accepts $L[a]$

A: accepts $L[a^*]$