PROPOSITIONAL LOGIC

A proposition is a statement that is either true or false

eg $2+2=4$ is a proposition, $x+2=4$ is not

TRUTH TABLE plots the outcomes for the inputs into the statement

p	q	$p \wedge q$	$p \vee q$	$p \rightarrow q$
1	0	0	1	0
1	1	1	1	1
0	1	0	1	1
0	0	0	0	1

NOT = $\neg$

OR = $\vee$

AND = $\wedge$

IF THEN = $\rightarrow$

IF ONLY IF = $\leftrightarrow$

EXCLUSIVE OR = $\oplus$

ORDER OF OPERATIONS = $\neg \wedge \vee \rightarrow \leftrightarrow$

- ☆ A statement is **CONSISTENT** if it is sometimes true.
- ☆ A statement is **INCONSISTENT** if it is never true (Also known as a contradiction)
- ☆ A **TAUTOLOGY** is a statement that is always true.
- ☆ A propositional **EQUIVALENCE** is two statements with the same truth table

COMPOUND STATEMENT

prepare the truth table for $(p \wedge q) \vee \neg r$

p	q	r	$(p \wedge q)$	$\neg r$	$(p \wedge q) \vee \neg r$
1	1	1	1	0	1
1	0	1	0	0	0
0	1	1	0	0	0

etc for each option

De morgans Law

1. $\neg(p \wedge q) \equiv \neg p \vee \neg q$
2. $\neg(p \vee q) \equiv \neg p \wedge \neg q$
3. $(p \rightarrow q) \equiv (\neg p \vee q)$
4. $p \rightarrow q \equiv \neg q \rightarrow \neg p$

PREDICATES + QUANTIFIERS

A statement such as $x > 3$ or $x = y + 3$ are not propositions until variables are defined. The predicate is the variable ~~eg x and~~ properties eg > 3 and x is the variable. The quantifier is the over arching result of the variable

We write such statements like this $P = > 3$ x = variable $P(x)$

eg $P(x) = x > 3$, $P(4)$ = TRUE and $P(2)$ = FALSE

You can assign quantifies to this statements such as

1. EXISTENTIAL $\exists$ some elements in set
2. UNIVERSAL $\forall$ all elements in set

eg $P(x) = x+1 > x$ domain is all real numbers. where:

$\forall x P(x)$ is True

eg $P(x) = x^2 > 10$ for all real numbers, since $5^2 > 10$ is false but $2^2 < 10$ is true we get

$\exists x P(x)$ is True

To negate such statements you do not have to provide the complete opposite. but simply that one example exist that isn't true. eg To negate the statement 'ALL MEN WEAR HATS' you simply need to show that 'SOME MEN DON'T WEAR HATS'

$\forall x P(x)$ = ALL MEN WEAR HATS

$\exists x \neg P(x)$ = SOME MEN DONT WEAR HATS

It is also worth noting that 'some men don't wear hats' is equivalent to 'not all men wear hats' eg

$\neg\forall x P(x) \equiv \exists x \neg P(x)$

Proofs

A proof is a connected sequence of logical statements that prove something is true.

While a computer can confirm if something is true or not within a range it cannot make the logical leap to make the statement that it is always true.

DIRECT PROOF is where you try to prove the statement directly to be true. For example in $P \rightarrow q$ is only ever false if q is false and p is true. If you can show that that never occurs then you have directly proved the statement.

eg if n is odd then n^2 is odd

let $n = 2k+1$ where k is an integer, any number multiplied by 2 is even, therefore adding 1 makes it odd.

$n^2 = (2k+1)^2 = 4k^2 + 4x + 1$

$= \underbrace{2(2k^2+2k)}_{\text{anything multiplied by 2 is even}} + 1$ ⇐ plus 1 at the end makes it odd

THEREFORE THE STATEMENT IS TRUE AND PROVEN DIRECTLY

INDIRECT PROOF otherwise known as

contraposithve proof. We first find a statement that has properchional equivalence eg $P \rightarrow q \equiv \neg q \rightarrow \neg p$. If you cannot prove the first statement is true then you try and poove the second statement is true.

example if n^2 is odd then n is odd, to prove indirectly we would try and show that if n^2 is even then n is even

eg let $n = 2k$. $n^2 = (2k)^2 = 4k^2$

$= 2(2k^2)$

↳ any number multplied by 2 is even

PROOF BY CONTRADICTION is where you

assume the opposite of your orginal statement. If you can prove the contradiction is false then the orginal statement is true.

eg the statement is $\sqrt{2}$ is irrational the opposite is that $\sqrt{2}$ is a rational number

A rational number is one that can be expressed as a fraction

$$\sqrt{2} = \frac{a}{b} \quad \text{or} \quad 2 = \frac{a^2}{b^2}$$

we can rearrange this to get $2b^2 = a^2$ since $2b^2$ is even (any number multplied by 2 is even. Then a^2 will be even, if a number is /2 it is not irrational

INDUCTION for certain statements

you may not automatically have a true or false statement eg $x+1>4$

Induction is used to show $\forall n\, P(n)$

1 BASIS STEP show that $P(1)$ is TRUE

2 INDUCTIVE STEP show if $P(k)$ is TRUE then $P(k+1)$ is TRUE

EXAMPLE prove $1+2+3+4\ldots n = \frac{n(n+1)}{2}$

1 BASIS STEP show $\frac{1(1+1)}{2} = 1$ this is TRUE

2 INDUCTIVE STEP we firstly assume that $1+2+3+4\ldots n = \frac{n(n+1)}{2}$

we can then show $n+1$ eg $1+2+3+4+n+(n+1) = \frac{(n+1)(n+1+1)}{2}$

by substituting the terms we then get

$$\frac{n(n+1)}{2} + (n+1) = \frac{(n+1)(n+1+1)}{2}$$

$$\frac{n(n+1)}{2} + \frac{2(n+1)}{2} = \frac{(n+1)(n+1+1)}{2}$$

$n^2 + 2n + 2 = n^2 + 2n + 2$ TRUE!!

COMBINATORIAL PRINCIPLES

PERMUTATIONS

The order matters - how many different ways can you arange something

eg 5 people in a queue = $5 \times 4 \times 3 \times 2 \times 1$ or $5!$ ways

Other examples can be out of 5 people how many pairs can you make in a queue?

eg $\frac{5!}{(5-2)!} = \frac{5 \times 4 \times 3 \times 2 \times 1}{3 \times 2 \times 1} = 5 \times 4$

➡ Permutation with repetition eg a lock combination. This is the most straight forward.

eg 4 digit lock combo with 9 numbers = $9 \times 9 \times 9 \times 9$

or 9^4

➡ Permutation without repetition eg top 3 finishes in a race of 10 people.

$\frac{10!}{(10-3)!} = \frac{10 \times 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1}{7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1} = 10 \times 9 \times 8$

examples: A password must be between 5-7 characters long and is made from the alphabet and numbers, must contain 1 letter.

5 CHARS Step 1: total selection $(26+10)^5$

Step 2: identify non qualifying string of all numbers 10^5

Step 3: $(36)^5 - (10)^5 = 60,366,176$ permutations

Step 4: Repeat for 6 and 7 digit passwords and add together

example: How many ways can MISSISSIPPI be arranged? The order of the repeating letters can be ignored

M I S P

I S P

I S

I

$4!\ 4!\ 2!$ $\quad \frac{11!}{4!4!2!}$

example: How many ways can you arrange $\{a, b, c, d, e, f, g\}$ containing the word 'bad'

STEP 1. reduce the set to $\{c, e, f, g\}$ and cycle 'bad' between the gaps

STEP 2: length 3 = 1 option 'bad'

STEP 3: length 4 = 4 (there is only one space going) $\times 2 = 8$

STEP 4: length 5 = 4×3 (2 spaces) $\times 3$

STEP 5: length 6 = $4 \times 3 \times 2$ (3 spaces) $\times 4$ etc

STEP 6: Add the options together to get the answer

COMBINATIONS

The order doesn't matter, but is still do with the arrangement of things

eg top 3 finishes (it doesn't matter the order)

COMBINATION WITH REPITITION → eg. scoops of ice cream you can have 3 scoops of ice cream how many combinations are there with 5 flavours

General formula $\frac{(r+n-1)!}{r!(n-1)!}$ where n = total to choose from

r = number of choices

you visualise it as follows:

Choc	Vani	Straw	Toffee	Pistac
0 scoops	0 scoops	1 scoop	0 scoop	0 scoop
		1 scoop		
		1 scoop		

there are $\frac{7!}{3!\text{ scoops }4!\text{ not selected}}$ boxes

eg $\frac{7!}{3!4!}$ combinations

COMBINATION WITHOUT REPITITION → is more straightforward. Treat it as a permutation and then divide it by the number of ways the final selection can be ordered eg top 3 finishes out of 100 runners

$\frac{100!}{(100-3)!} = 100 \times 99 \times 98$ the 3 runners can be arranged in 3! ways so the answer is

$(100 \times 99 \times 98) / 3!$

example: How many binary strings of length 8 contain equal numbers of 1 and 0?

Does the order matter = no, so it is a combination

Do the items repeat = yes but see logic below

We have 4 1's and their are 8 available positions if order mattered we would arrange these $8 \times 7 \times 6 \times 5$ ways. As order doesn't matter we divide it by $4 \times 3 \times 2 \times 1 = 70$ ways

Pidgeon Hole Theory

If you have K boxes and K+1 items then one box must contain at least 2 items $\lceil N/K \rceil$ (where $\lceil\ \rceil$ = round up, $\lfloor\ \rfloor$ round down

eg If there are more than 367 students in a class then at least 2 have the same birthday

eg If you pick 7 countries at random at least 2 are on the ~~continent~~ same continent

eg If we a deck of cards how many must you pick to get four of the same suit?

= you can pick 3 cards from each suite (3×4)

but the moment you pick the 13th you have 4.

AUTOMATA theory

A set is a collection of unordered and distinct objects called elements.

IMPORTANT
1. There is no order in a set
2. Repeated elements are listed once
3. Sets can be finite or infinite

IMPORTANT SYMBOLS
$\in$ = inclusion eg $\emptyset \in A$ "$\emptyset$ is an element in A"
$\notin$ = not an element eg $30 \notin A$ "30 isn't in A"
$|B|$ = number of elements in the set
Σ = An alphabet
ξ = An empty string
$\emptyset$ = empty set

The symbol Σ^* is all of the strings that can be made from the language Σ eg. $\Sigma = \{0, 1\}$ $\Sigma^* = \{\xi, 0, 1, 00, 01, 10, 11, 111...\}$

If we don't care about the empty string ξ then we write it as Σ^+

Sometimes we might be interested only in strings of a particular length. If we write this as Σ^k eg. $\Sigma^2 = \{00, 11, 01, 10\}$

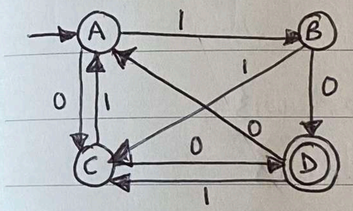
$\Sigma^3 = \{000, 111, 101, 011, 110...\}$

using permutations for repeating selections we would say that there are $2 \times 2 \times 2$ or 2^3 permutations $\Sigma^k = |\Sigma|^k$

What is an automaton?

An automaton is a simplistic computer with limited memory. A 5 tuple automaton has the following components

1. States Q a finite set
2. Alphabet Σ a finite set
3. Transitions δ $Q \times \Sigma \to Q$ (state + alphabet moves to another state)
4. Starting state q_0 where $q_0 \in Q$
5. Accept state F where F is a subset $\subseteq$ of Q



In this example $Q = \{A, B, C, D\}$

$\Sigma = \{1, 0\}$

$q_0 = A$

$F = \{D\}$

$\delta =$

	0	1
A	C	B
B	D	C
C	D	A
D	A	C

For the string to be accepted it must end on accept state

eg $110 = A \xrightarrow{1} B \xrightarrow{1} C \xrightarrow{0} D$

The set of all the strings accepted by the automaton is called the LANGUAGE. If M has an alphabet Σ then $L(M)$ is the language

$$L(M) = \{\underbrace{x \in \Sigma^*}_{\text{strings}} \mid \underbrace{M \text{ accept } x}_{\text{criteria}}\}$$

Deterministic Finite Automaton

A DFA is an automaton where

- For each state there is one transition ONLY
- There is a unique starting state

If either state is not met then it is considered non-deterministic.

LANGUAGES

operations on sets can be done as follows:

UNION $A \cup B = \{x \mid x \in A, x \in B\}$

eg $A = \{red, green, pink\}$ $B = \{apple, banana, kiwi\}$

$A \cup B = \{red, green, pink, apple, banana, kiwi\}$

1. $A \cup B = B \cup A$
2. $(A \cup B) \cup C = A \cup (B \cup C)$
3. $A \cup \emptyset = A$
4. $A \cup A = A$

CONCATENATION $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$

eg $A \circ B = \{redapple, redbanana, redkiwi, greenapple, greenbanana...\}$

1. $(A \circ B) \circ C = A \circ (B \circ C)$
2. $A \circ \varepsilon = \varepsilon \circ A = A$
3. $A \circ \emptyset = \emptyset$

combining operations union and concatenation

1. $(A \cup B) \circ C = (A \circ C) \cup (B \circ C)$
2. $A \circ (B \cup C) = (A \circ B) \cup (A \circ C)$

kleene star operations

1. $\emptyset^* = \emptyset$ empty language = regular expression
2. $\varepsilon^* = \varepsilon$
3. $(A^*)^* = A^*$
4. $A^* A^* = A^*$
5. $(A \cup B)^* = (A^* B^*)^*$

Compound Regular Expressions

CONCATENATION = If R_1 and R_2 are regular expressions then $R_1 \circ R_2$ is RegEx

UNION = If R_1 and R_2 are RegEx then $R_1 \cup R_2$ is also

KLEENE STAR = If R is RegEx then R^* is also

examples $ab^* = \{a, ab, abb, abbb...\}$

$ab^* \cup b^* = \{a, ab, abb\} \cup \{\varepsilon, b, bb\} = \{\varepsilon, a, b, ab, bb, abb...\}$

$ab^+ \cup b^+b = \{ab, abb, abb...\} \cup \{bb, bbb\} = \{ab, bb, abb...\}$

$= ab^* \cup b^* / \{a, \varepsilon, b\}$ eg the same as but not including

Examples of a binary alphabet $\Sigma = \{a, b\}$

1. $\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb \ldots\}$
2. $\Sigma^* a = \{a, aa, ba, aaa, aba, baa \ldots\}$ the same as above ending with a
3. $\Sigma^* a \Sigma^* = \{a, aa, ba, aaa \ldots\}$ ~~$= A^* A^*$ is just A^*~~

ORDER OF OPERATIONS *, CONCATENATION, UNION

eg $a \cup bc^* = a \cup b(c^*) = a \cup (b(c^*))$

$\{a, b, bc, bcc, bccc, \ldots\}$

Identifying a regular expression for a string:

$\{bb, abb, bba, bbb, aabb, abba, abbb, babb\}$

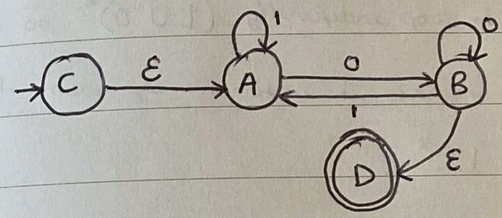
1. We can see "bb" repeats through each string
2. It can have an empty string, b or a in front of it
3. It can have an empty string, b or a behind it

$(a \cup b)^* bb (a \cup b)^*$ or $\Sigma^* bb \Sigma^*$

KLEENES THEORUM ⇨ A language is regular only if it can be described by a regular expression

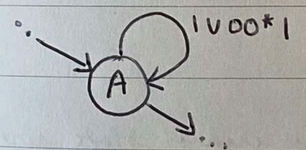
Regex to Finite Automaton

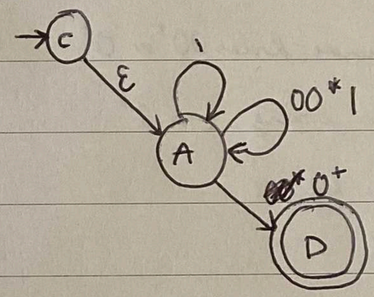
If there is an automaton (finite) it can be represented by a regular expression



Consider the automaton to the right to find the regex we start to remove transitions.

Consider state B, the path always comes from A with a 0, it then may loop 0 a number of times before out putting a 1 or ε.

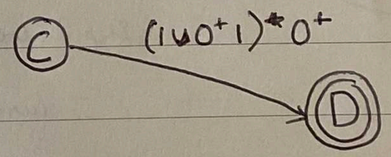




the loops going around A can just be removed to $1 \cup 00^* 1$

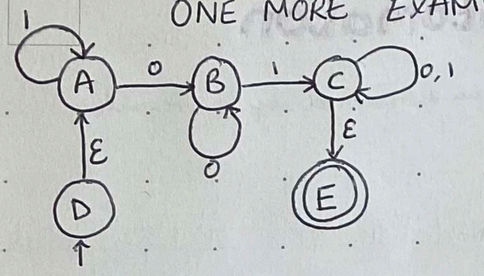
00^* is also written as 0^+

we then consider the transition from C → A → D



and combine the ingoing, outgoing and looping expressions.

ONE MORE EXAMPLE! FINITE AUTOMATON REGEX

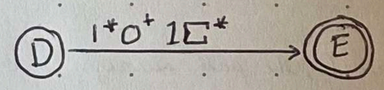


step 1 → look at removing C.

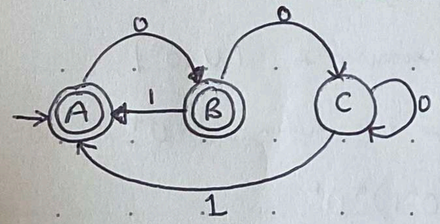
you start with a 1 then we either have 1 or a 0 to loop indefinitely $(1 \cup 0)^*$ so we can replace it with the expression $1(1 \cup 0)^*$ or $1\Sigma^*$

We then look at removing B which is an incoming 0 with 0 looping originally hence 00^* or 0^+ and then A is a looping 1 or 1^* we combine these to get

$1^*0^+1\Sigma^*$

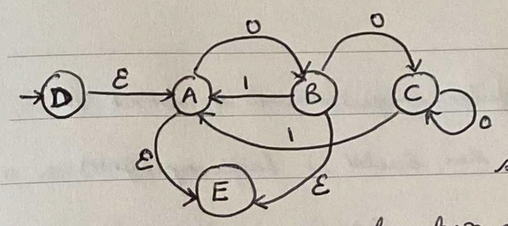


EXAMPLE 2 ⇨ MULTIPLE FINAL STATES

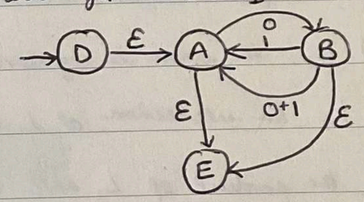


Step 1 create a new initial state with an ε transition to A

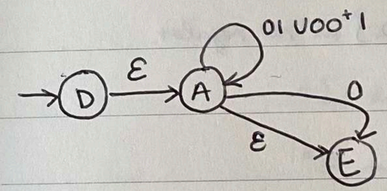
Step 2 create a final state connecting two current final states using ε



we first look at removing state C. It has an incoming transition of of 0, loops 0, then outputs on 1 so we get 00^*1 or 0^+1



Then we look at removing B. This must cover the transition from B→E and B→A. For B→A we have an incoming connection of 0 and a return of 1 and 0^+1 so we get $01 \cup 00^+1$



We then remove A to get

$(01 \cup 00^+1)^*(\varepsilon \cup 0)$

PUMPING LEMMA

is a way of helping to prove if a language is regular or not.

A regular language is one that can be

- ★ Accepted by finite Automaton
- ★ Can be accepted by a regular expression.
- ★ Every finite language is regular

one way of identifying regular languages is to break it down into its component parts and then build it back up again

CLOSURE PROPERTIES

- ☆ $L_1 \cup L_2$. the union is regular if L_1 and L_2 are regular
- ☆ $L_1 \cap L_2$. the intersection of L_1 and L_2.
- ☆ $L_1 L_2$., the product of L_1 and L_2
- ☆ L^* ., the kleene star of L_1.
- ☆ $U - L_1$ $(\Sigma^* - L_1)$. or the complement of L_1.

example Prove $L = \{x \in \{a,b\}^* \mid \#a\ in\ x = \#b\ in\ x\}$ is not regular.

$L = \{ab, aabb, abab, abba\}$

STEP 1 - Assume that L is regular

STEP 2 - We know $L' = \{x \in a^* b^*\}$ is regular

STEP 3 - If L' and L is regular then $L \cap L'$ is regular

STEP 4 - $L_1 \cap L = \{a^n b^n . n \in \mathbb{N}\}$ starts with an a, then n a, then n b^s.

STEP 5 = $\{a^n b^n \mid n \in \mathbb{N}\}$ is not an regular language therefor L is not regular

The PUMPING LEMMA theory states that all regular languages have a special property.

This special property is all strings in the langage can be pumped if they are longer or equal to the PUMPING LENGTH. eg a string has a section that when repeated and still be in the language

GRAMMAR

⇨ is a set of rules for connecting strings

⇨ another way of representing languages

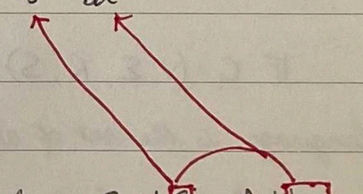
⇨ 4 tuple (V, Σ, R, S)

- V variables, a finite set of symbols
- Σ terminals, a finite set of letters
- R rules, a finite set of mappings. Each rule being a variable and a string of variables and terminals.
- S start Variable, a member of V

example S = non terminal rules ⇒ S = bSa

a, b = terminals ⇒ S = ba

S = start variable



we can generate strings by substitution S = bSa ⇨ bbaaa

S = bSa ⇨ bbSaa ⇨ bbbaaa etc

Example 1 $S \to aS \mid T$, $T \to b \mid \epsilon$ } first 3 strings derived from S.

- ★ Variables are S and T
- ★ Terminals are a and b.
- ★ Starting variable S

$S \Rightarrow aS$ since $S \Rightarrow T$ we can do $S \Rightarrow aS \Rightarrow aT$. since $T \Rightarrow$ empty $\in$. $S \Rightarrow a$

$S \Rightarrow aS$ since $S \Rightarrow T$ and $T \Rightarrow b$ we can do $S \Rightarrow aS \Rightarrow aT \Rightarrow ab$

$S \Rightarrow T$ since $T \Rightarrow \epsilon$ we can do $S \Rightarrow T \Rightarrow \epsilon$

Example 2 show aabba is derived from. $S \to aTS \mid aS \mid \varepsilon$, $T \to SbT \mid ba$

$S \Rightarrow aTs$ replace T with SbT $S \Rightarrow aTS \Rightarrow aSbTs$ then replace S by aS to get $S \Rightarrow aTS \Rightarrow aSbTs \Rightarrow aaSbTs$ then replace S with epsilon then T with ba then S with epsilon to get

$S \Rightarrow aTs \Rightarrow asbTS \Rightarrow aaSbTs \Rightarrow aabTs \Rightarrow aabaS \Rightarrow aaba$

LANGUAGE OF GRAMMAR

IF $G = (V, \Sigma, R, S)$ the $L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$

eg The language is the set of all words in sigma star such that w can be derived from S

example 1 What is the language of the following grammar?

$S \to bSa$

$S \to ba$

Step 1 - what does it look like? ba, bbaa, bbbaaa, bbbb aaaa

$L(G) = \{b^n a^n \mid n > 1\}$

Therefore $G_1 = (\{S\}, \{a, b\}, R, S)$, $R = \{S \to bSa, S \to ba\}$

variable | language | starting variable | definition of R

BUILDING A GRAMMAR

EXAMPLE: Binary strings with an even number of 0's

★ STEP 1 : If the first letter is 1 then it must be followed by an even number of 0's. We shall call these A, So $S \to 1A$ where A = even number of 0's

★ STEP 2 : If the first letter is 0, then it can have strings before and after it, but it must contain another 0 so $S \to 0A0B$ where A and B = even number of 0s

★ We then replace A and B with S to get

$S \to 1S \mid 0S0S \mid \varepsilon$

EXAMPLE: All strings of the form 0^+1^+

STEP 1: All strings starting with 0^+ are $U \to 0U|0$

STEP 2: All strings starting with 1^+ are $V \to 1V|1$

we compare these to say:

$S \to UV$ $\{0, 1, 00, 11, 000, 111, 0000, 1111...\}$

$U \to 0U|0$ $\{0, 00, 000, 0000...\}$

$V \to 1V|1$ $\{1, 11, 111, 1111...\}$

EXAMPLE - Design a context free grammar for $\{a^m b^n | n \geq m\}$

STEP 1 - decompose the string, if n is greater than or equal to m we can say it will be $n-m$. so

$$a^m b^n = a^m \times b^{n-m} \times b^m$$

STEP 2 - if $n-m=0$ then we get $a^m b^m$. this can be written as

$S \to aSb|\varepsilon$ $\{ab, aabb, aaabbb\}$

STEP 2 - if $n-m>0$ then we get b^i (where $i = n-m$)

$U \to bU|b$ $\{b, bb, bbb\}$

combining these both we get $S \to aSb|V|\varepsilon$ $\{ab, abbb, aabb$ etc$\}$

$U \to bU|b$

CONTEXT-FREE GRAMMAR

CHOMSKY NORMAL FORM is where

1. Every rule is of the form $S \to XU$

   $S \to a$

2. Where a is any terminal
3. Where X, U, S are non-terminals
4. X and U are not the start variables
5. $S \to \varepsilon$ is permitted if S is the start variable

example that is not chomsky normal form is

1. $S \to 1S|0S0S|\varepsilon$, because S also appears on the right
2. $V \to \varepsilon$, if V is not the S variable it cannot go to ε
3. $U \to V$, where V is a variable
4. $X \to 1UV$, where the length of the rule is ≥ 3

you can convert something into a chomsky normal form by doing the following

1. $S \to 1S|0S0S|\varepsilon$ becomes $S_0 \to S$, $S \to 1S|0S0S|\varepsilon$ — S_0 becomes the start variable
2. $V \to aU$ and $U \to a\varepsilon$ becomes $V \to aU|a$ and $U \to a$ and ε is eliminated