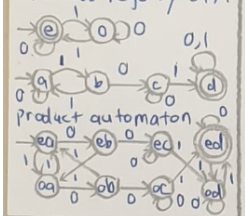
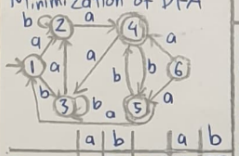
NGUYEN XUAN BINH Id: 887799
Theory Of Computation

L2.1: L regc by DFA ⟺ $\bar{L}$ regc by a DFA ⇒ RL are closed under complementation (swith q_c and non q_c)

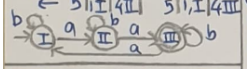
L2.2: A, B regc by DFA ⇒ $A \cap B$, $A \cup B$ are also regc by DFA



Minimization of DFA

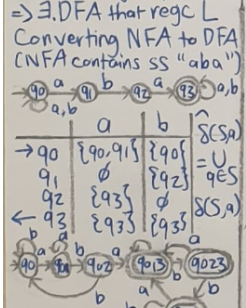


	a	b		a	b
I: → 1	2,I	3,I	I→1	2II	3I
2	4,II	2,I	3	2II	3I
3	2,I	3I	II: 2	4III	2II
II ← 4	3I	5II	III←4	3I	5III
← 5	1,I	4II	5	1,I	4III



T3.2 L regc by NFA ⇒ ∃ DFA that regc L

Converting NFA to DFA (NFA contains ss "aba")



	a	b	$\hat{\delta}(S,a)$
→ q0	{q0,q1}	{q0}	$= \bigcup_{q \in S} \delta(S,a)$
q1	∅	{q2}	
q2	{q3}	∅	
← q3	{q3}	{q3}	

T3.3: L regc by ε-A ⇒ ∃ NFA that regc L

D4.1: Syntax of Regex
1. ϕ & ε are regex over Σ
2. a is regex over Σ if $a \in \Sigma$
3. If r & s are regex over Σ ⇒ $r \cup s$, $r \cap s$, r^* are also regex over Σ

$r(s \cup t) = rs \cup rt$
$r \cup r = r$, $r \cup \phi = r$
$\varepsilon r = r$, $\phi r = r$, $r^* = \varepsilon \cup r^* r$

D4.3: L is regular if it can be described by regex

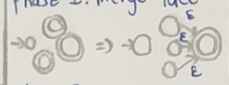
T4.1: L regc by regex are also regc by DFA

T4.2: L regc by DFA can be described with regex ⇒ DFA ⟺ Regex

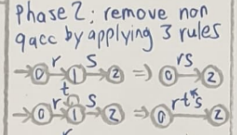
- Regex contains 0110 or 1001 or both as ss
$(0 \cup 1)^*(0110 \cup 1001)(0 \cup 1)^*$
- Regex contains a^n, n divide by 2 or 3 or both
$(aa \cup aaa)^* (\varepsilon \cup aa \cup aaa \cup aaaaa)$

Methods of converting DFA into regex

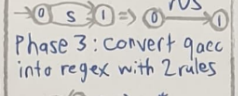
Phase 1: merge q_{acc}



Phase 2: remove non q_{acc} by applying 3 rules



Phase 3: convert q_{acc} into regex with 2 rules



$(r_3 \cup r_4 r_1^* r_2)^*$

L4.3: Pumping Lemma for regular L (RL)

Let L be a RL. Then $\exists p \geq 1$ such that $\forall x \in L$ with $|x| \geq p$ can be divided into 3 parts: $x = uvw$ satisfying 3 conditions
(i) $|uv| \leq p$ (ii) $|v| \geq 1$ (iii) $uv^i w \in L \ \forall i \geq 0$

Usage of pumping lemma if L is RL ⇒ L satisfies pumping lemma. If L not satisfy pumping lemma ⇒ L is not RL

* Pumping lemma can't be used to prove L is RL, but only L is not RL. To prove L is RL, try to find DFA, NFA or regex

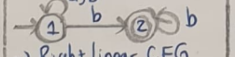
- Prove $a^k b^k$, $k \geq 0$ not a RL
$x = a^p b^p \in L$, $p \geq 1$
$|x| = 2p > p$, $x = uvw$
$u = a^m$, $v = a^n$
$w = a^{p-(m+n)} b^p$
$|uv| \leq p \Rightarrow m+n \leq p$
$|v| \geq 1 \Rightarrow n \geq 1$
Pumping v 0 times
$uv^0 w = a^m a^{p-(m+n)} b^p$
$= a^{p-n} b^p \notin L$ as $n \geq 1$
⇒ contradict ⇒ L not RL
$m \geq n$ like $m > n$; uv^2w $m \leq n$ like $m < n$; uv^2w
- Prove $a^m b^n$, $m < n$ is not a RL
$x = 0^p 1^{p+1}$, $p \geq 1$
$p < p+1 \Rightarrow x \in L$
Also $|x| = 2p+1 > p$
$u = 0^p 1^{p-n-m}$, $v = 1^n$
$w = 1^{m+1}$, $m+n \leq p, n \geq 1$
Pump v 0 times
$\Rightarrow uv^0 w = 0^p 1^{p-n-m} 1^{m+1}$
$= 0^p 1^{p+1-n} \notin L$ as $n \geq 1$
- Prove $a^m b^n$, $m > n$
$x = 0^{p+1} 1^p$, $p \geq 1$
$p+1 > p \Rightarrow x \in L$
Also $|x| = 2p+1 > p$
$u = 0^{m+1}$, $v = 0^n$
$w = 0^{p-n-m_1} 1^p$, $m+n \leq p$, $n \geq 1$
Pump v 0 times
$\Rightarrow uv^0 w = 0^{m+1} 0^{p-n-m_1} 1^p$
$= 0^{p+1-n} 1^p \notin L$ as $n \geq 1$

D5.2: L is context-free if it can be generated by some CFG

T5.1: The class of CFG is closed under union, concatenation and star
L_1, L_2 is CFG ⇒ $L_1 \cup L_2$, $L_1 L_2$ and L_1^* are CFG

- $w \in \{a,b\}^*$, $n_a = n_b$
$S \to SS \mid aSb \mid bSa \mid \varepsilon$
- $w \in \{a,b\}^*$, $n_a \neq n_b$
$S \to A \mid B$
$A \to EaA \mid EaE$
$B \to EbB \mid EbE$
$E \to aEbE \mid bEaE \mid \varepsilon$
- $w \in \{a,b\}^*$, $n_a > n_b$
$S \to Aa \mid MS \mid SMA$
$A \to Aa \mid \varepsilon$
$M \to MM \mid aMb \mid bMa \mid \varepsilon$
- $a^m b^n$, $m > n$
$S \to aSb \mid aS \mid a$
- $a^m b^n$, $m \neq n$
$S \to AT \mid TB$
$A \to Aa \mid a$ $B \to Bb \mid b$
$T \to aTb \mid \varepsilon$
- $\{ucv \mid u, v \in \{a,b\}^*$ and $|u| = |v|\}$
$S \to aSa \mid bSb \mid aSb \mid bSa \mid c$
- $\{a^i b^j c^k \mid i, j, k \geq 0$ $i+j=k\}$
$S_0 \to aS_0c \mid S_1$
$S_1 \to bS_1c \mid \varepsilon$
- Palindrome
$S \to aSa \mid bSb \mid a \mid b \mid \varepsilon$

T5.2: each RL can be generated by some right linear CFG



⇒ Right linear CFG
$A_1 \to aA_1 \mid bA_1 \mid bA_2$
$A_2 \to bA_2 \mid \varepsilon$

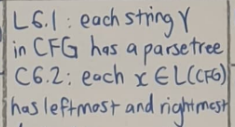
For each q_{acc} there is a rule $A_{q_{acc}} \to \varepsilon$
For each $q \to q'$ ⇒ rule $Aq \to aAq'$

T5.3: if a lang can be generated by right linear CFG ⇒ L is RL
From T5.2 & T5.3 ⇒ Right linear CFG ⟺ RL

L6.1: each string Y in CFG has a parse tree

C6.2: each $x \in L(CFG)$ has leftmost and rightmost derivation

Parse trees



$E \to T \mid E+T$
$T \to F \mid T*F$
$F \to a \mid (E)$
$E \to E+T \to T+T \to F+T \to a+a$

Ambiguous & unambiguous CFG: a CFG is ambiguous if $\exists x \in L(CFG)$ having 2 different parse trees
Otherwise CFG is unambi~
- A context-free lang for which all grammars are ambi is inherently ambi L
- Ex: $\{a^i b^j c^k, i=j$ or $j=k\}$ is inherently ambiguous but still CFG

Left recursion: can derive some variable A with ≥ 1 steps the string Aa, $a \in (VUZ)^*$
$E \to E+T$ is left recur in a single step
$S \to ASa \mid b$
$A \to BB \mid dA$ $B \to b \mid \varepsilon$
is left recursive: $S \to ASa \to BBSa \to BSa \to Sa$

Chomsky Normal Form
It's a CFG satisfying
- no vars except start S are nullable
- Besides $S \to \varepsilon$, all productions are:
$A \to BC$ or $A \to a$
or $A \to BC \mid DE \mid \ldots \mid a \mid b \ldots$
- Start S doesn't occur an RHS of any production

Variable A is nullable in CFG if $\exists G, A \Rightarrow_G \varepsilon$

T7.3: For all CFG, we can construct an equivalent CNF
Step 1: if S in RHS, create new $S' \to S$
Step 2: remove ε productions
Step 3: remove unit productions

$S \to AB \mid c$
$A \to T \mid aA$
$B \to TT \mid \varepsilon$
$T \to bS$

⇒

$S \to AB \mid YD \mid XA \mid c$
$D \to AB \mid YD \mid XA \mid c$
$A \to YD \mid XA$
$B \to TT \mid YD$ $T \to YD$
$X \to a$ $Y \to b$

CYK algorithm: decide if abba belongs to the CFG: $A \to BA \mid a$, $S \to AB \mid BA$, $B \to AB \mid b$

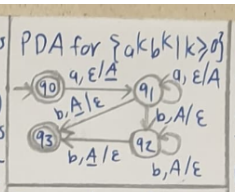
Ni,k	1:a	2:b	3:b	4:a
1	A	B	B	A
2	S,B	ϕ	S,A	
3	ϕ	S,A		
4	S,A			

S in $N_{1,4}$ ⇒ abba belongs
$N_{1,4} = N_{11} \times N_{23} + N_{12} \times N_{32} + N_{13} \times N_{41}$

Push-down automaton PDA

T7.4: L is context free ⟺ regc by NPDA
A CFL is deterministic if it can be regc by a deterministic PDA
$\delta(q, b, \gamma) = (q_1, \gamma_1)$
- $b = \varepsilon \Rightarrow$ can transition on
- $\gamma = \varepsilon \Rightarrow$ push
- $\gamma \neq \varepsilon$, $\gamma_1 = \varepsilon \Rightarrow$ pop

PDA for $\{a^k b^k \mid k \geq 0\}$

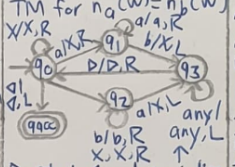


Pumping lemma for CFL
Let L be CFL. For $n \geq 1$ $\forall z \in L$, $|z| \geq n$ can be divided into $uvwxy$ s.t
1. $|vx| \geq 1$ & $uv^i wx^i y$
2. $|vwx| \leq n$ $\forall i \geq 0$

Prove $L = \{a^k b^k c^k \mid k \geq 0\}$ is not context-free
Let $z = a^n b^n c^n \in L$
$z = uvwxy$, $|vx| \geq 1$
$|vwx| \leq n$, $n \geq 1$
- 2nd cond ⇒ vx has at least one symbol: a or b or c
- 3rd cond ⇒ vx can't contain every sym a, b, c
⇒ $uv^0 wx^0 y$ cannot contain equal num of all a, b, c ⇒ L is not CF

Turing Machine (TM)
$\delta(q, a) = (q', b, \Delta)$
In state q and read a ⇒ write b & move to q'
$\Delta = R, L$, Stationary

TM for $n_a(w) = n_b(w)$



▷: start ◁: end rewinding

L8.1 & L8.2: L regc by multi-track and multi tape TM can also be regc by a standard TM

T8.3: if L can be reg by nondeter TM ⇒ it can be regc with deter TM

- Decidable / Turing-decidable / recursive
- Semi-decidable / Turing-recognizable / recursively enumerable
- Undecidable / non recursively enumerable

- Chomsky's hierachy
Type 3: RL, DFA, regex, NFA, right & left linear G
Ex: $\{a^k \mid k \geq 0\}$
Type 2: CFG, CFL, PDA, NPDA, $\{a^k b^k, k \geq 0\}$
Type 1: CSG, CSL, linear-bounded automata $\{a^k, b^k c^k \mid k \geq 0\}$
Type 0: unrestricted G, SDL, Turing machine
U: universal language

- L9.1: let A, B be DL ⇒ $\bar{A}$, $A \cup B$ and $A \cap B$ are also DL
- L9.2: let A, B be SDL ⇒ $A \cup B$, $A \cap B$ are SDL
- L9.3: A and $\bar{A}$ are SDL ⟺ A is DL

*C9.4: Let A be SDL but not DL ⇒ $\bar{A}$ is not SDL

T9.6: family of all langs over all Σ is uncountable

$U = \{c_M w \mid$ TM accepts $w\}$

T9.8: U is SDL

T9.9: U is not DL

C9.10: $\bar{U}$ is not SDL

$D = \{c \in \{0,1\}^* \mid c \notin L(c_M)\}$

T9.7: D is not SDL
- Halting problem
$H = \{c_M w \mid$ TM halts on $w\}$

T10.1: H is SDL but not DL

C10.2: $\bar{H}$ is not SDL
- Non-emptiness prdo
$NE = \{c \in \{0,1\}^* \mid L(c_M) \neq \phi\}$

T10.3: NE is SDL but not DL

Reduction proof
L10.8: If $A \leq_m B$ and B is DL ⇒ A is DL
C10.9: If $A \leq_m B$ & A is not DL ⇒ B is also not DL
Usage: choose a known UDL A → do reduction from A to B → B is UDL
Ex: prove NE is UDL
* UDL can still be SDL we design a reduction mapping f from U which is UDL to NE. Given arbitrary $c_M w$ to U, reduction produces string $f(c_M w) = c_{M'}$ such that $c_M w \in U \Leftrightarrow c_{M'} \in NE$
⇒ if NE can be solved, so is universal lang ⇒ X

Rice Theorem: all non-trivial semantic props S of TM are undecidable
- RE: family of all SDL
- Semantic S of TM is any fam of SDL ⇒ $S \subseteq RE$
- TM has S if $L(M) \in S$
- trivial S: $S = \phi$ (no TM has S) and $S = RE$ (all TM have this prop)
- Prop S is decidable if codes $(S) = \{c_M \mid M$ is TM and $L(M) \in S\}$ is decidable

Prove NE is UDL by Rice
Semantic prop: $NE = \{L \in RE / L \neq \phi\}$. This prop is non-trivial because: $NE \neq \emptyset$ (find SDL $\neq \emptyset$), $NE \notin RE (\phi \in RE \setminus NE)$
⇒ By Rice, codes (S) is UD

T11.2 & T11.3: L generated by unrestricted G ⟺ L regc by a TM

T11.4: L generated by context-sensitive G ⟺ L regc by linear-bounded automata

$a^k b^k c^k, k \geq 0$
$S \to aBSc \mid \varepsilon$
$Ba \to aB$
$Bc \to bc$
$Bb \to bb$

27 Oct 2021

1. (a) Design a deterministic finite automaton that recognises the language

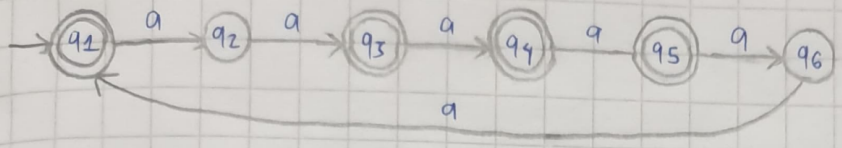
$$L = \{a^n \mid n \text{ is divisible by 2 or 3 (or both)}\}.$$

(The value 0 is considered to be divisible by any number.) 8 points

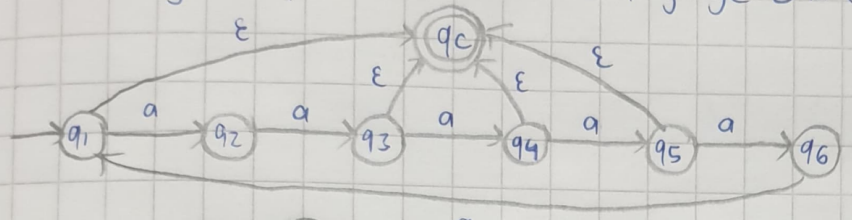
(b) Give a regular expression that describes the language L in part (a). 6 points

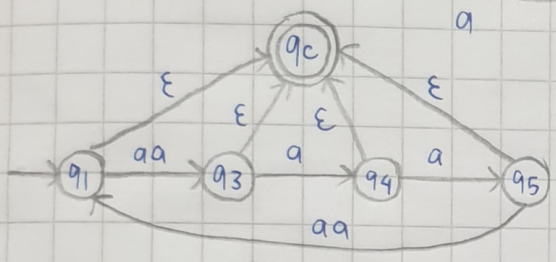
Exercise 1: Design DFA for

a) $L = \{a^n \mid n$ is divisible by 2 or 3 or both$\}$

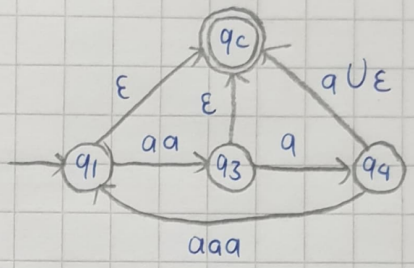


b) Give a regular expression that describes language L in (a)

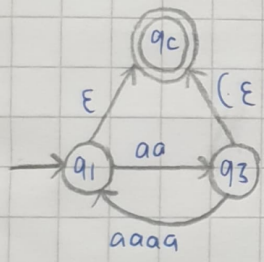




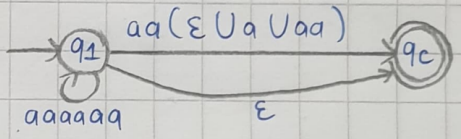
⇒



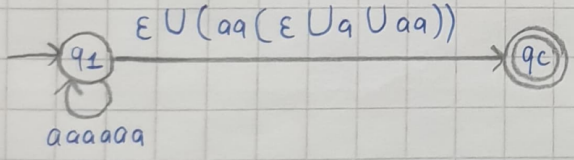
⇒



⇒



⇒



⇒ Regex of the language : $(aaaaaa)^*(\varepsilon \cup (aa(\varepsilon \cup a \cup aa)))$

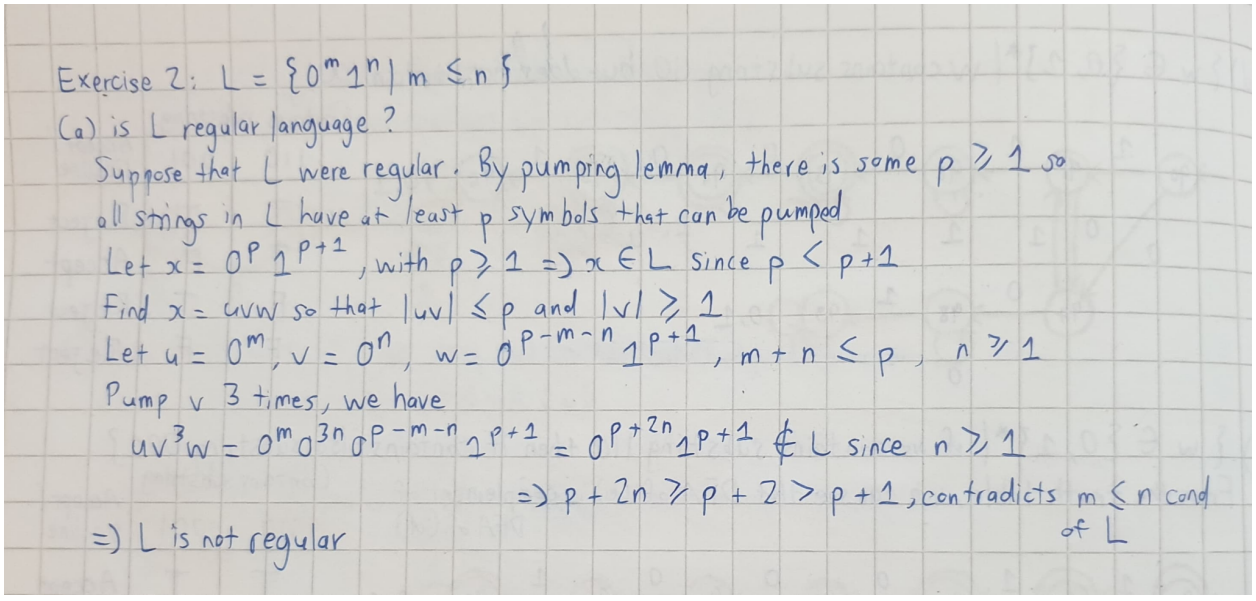
or $(aaaaaa)^*(\varepsilon \cup aa \cup aaa \cup aaaa)$

2. Is the language

$$L = \{0^m 1^n \mid m \leq n\}$$

(a) regular, (b) context-free but nonregular, (c) decidable but not context-free, (d) undecidable? Give a *precise justification* for your answer. 15 points

(a) The language is not regular
Justification:



(b) True. We can devise a context free grammar for the problem above and we have already proved in a that it is not regular
The CFG for the language
S -> 0S1 | S1 | e
(c) False. This problem is both decidable and context-free
(d) False. Since there exists a CFG that can generates the language in (b), we know for sure that it is a decidable problem

3. A language class $\mathscr{C}$ is said to be *closed under complement*, if for every $L \in \mathscr{C}$ also $\bar{L} \in \mathscr{C}$.

(a) Show that the class of regular languages is closed under complement. 7 points

(b) Show that the class of context-free languages is not closed under complement. (*Hint:* You may use the information from the course that the language $L = \{a^n b^n c^n \mid n \geq 0\}$ is not context-free.) 8 points

(a) The class of regular languages is closed under complement proof

Proposition 3. *Regular Languages are closed under complementation, i.e., if L is regular then $\overline{L} = \Sigma^* \setminus L$ is also regular.*

Proof.
- If L is regular, then there is a DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L = L(M)$.
- Then, $\overline{M} = (Q, \Sigma, \delta, q_0, Q \setminus F)$ (i.e., switch accept and non-accept states) accepts $\overline{L}$. □

(b) The class of context-free languages is not closed under complement

4. Which of the following claims are true and which are false? (No proofs are needed, just indicate your choice by the letter T or F.)

   (a) The union of any two regular languages is context-free.

   (b) Every language that can be recognised by a nondeterministic pushdown automaton can be generated by a context-free grammar.

   (c) Every language that can be recognised by a deterministic pushdown automaton can be described by a regular expression.

   (d) There exist undecidable context-free languages.

   (e) Nondeterministic Turing machines recognise ("semidecide") exactly the recursively enumerable languages.

   (f) The language $\{a^n b^n \mid n \geq 0\}$ can be recognised by a nondeterministic finite automaton.

   (g) The complement of every decidable languages is semidecidable.

   (h) The computation of a deterministic Turing machine terminates on every input.

16 points

Total 60 points

(a) The union of any two regular languages is context-free
(b) Every language that can be recognised by a nondeterministic pushdown automaton can be generated by a context-free grammar
(c) Every language that can be recognised by a deterministic pushdown automaton can be described by a regular expression
(d) There exist undecidable context-free languages
(e) Nondeterministic Turing machines recognise ("semidecide") exactly the recursively enumerable languages
(f) The language ($a^n b^n$ | n >= 0} can be recognised by a nondeterministic finite automaton
(g) The complement of every decidable languages is semidecidable
(h) The computation of a deterministic Turing machine terminates on every input.

14 April 2021

1. Design **deterministic** finite automata for recognising the following languages:

   (a) $\{w \in \{0,1\}^* \mid w$ contains the substring 110$\}$,

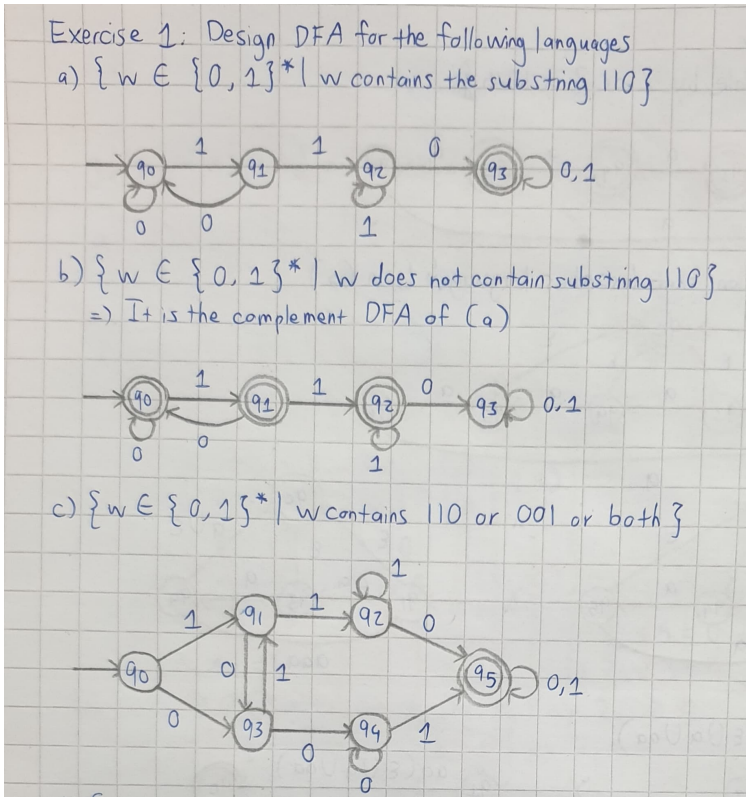
   (b) $\{w \in \{0,1\}^* \mid w$ does **not** contain the substring 001$\}$,

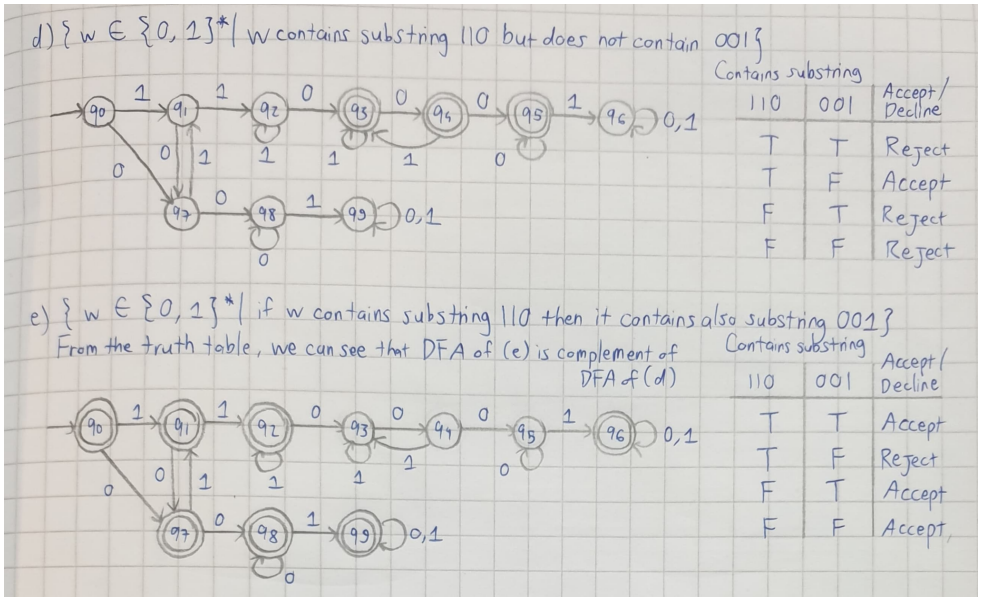
   (c) $\{w \in \{0,1\}^* \mid w$ contains the substring 110 **or** contains the substring 001 (or both)$\}$,

   (d) $\{w \in \{0,1\}^* \mid w$ contains the substring 110 **but** does **not** contain the substring 001$\}$,

   (e) $\{w \in \{0,1\}^* \mid$ **if** w contains the substring 110 **then** it contains also the substring 001$\}$.

15 points





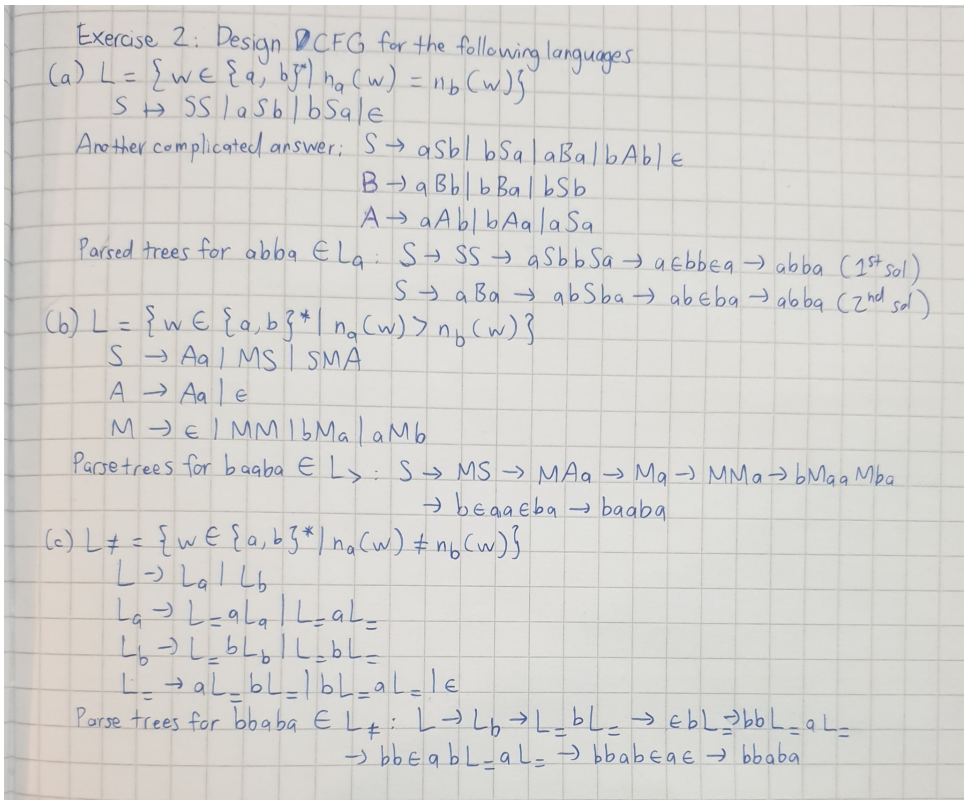
2. For a string $w \in \{a,b\}^*$, let us denote by $n_a(w)$ the number of a's in w and by $n_b(w)$ the number of b's in w. Design context-free grammars for the following three languages:

   (a) $L_= = \{w \in \{a,b\}^* \mid n_a(w) = n_b(w)\}$,

   (b) $L_> = \{w \in \{a,b\}^* \mid n_a(w) > n_b(w)\}$,

   (c) $L_\neq = \{w \in \{a,b\}^* \mid n_a(w) \neq n_b(w)\}$.

   Give also parse trees in your grammars for the following strings: (a) $abba \in L_a$, (b) $baaba \in L_>$, (c) $bbaba \in L_\neq$. (*Hints:* In part (b), it might be helpful to first design a grammar for the language $L_\geq = \{w \in \{a,b\}^* \mid n_a(w) \geq n_b(w)\}$. In part (c), notice that for any two integers k and l, $k \neq l$ if and only if $k > l$ or $l > k$.) 15 points


Exercise 2: Design a CFG for the following languages

(a) $L = \{w \in \{a,b\}^* \mid n_a(w) = n_b(w)\}$

$S \mapsto SS \mid aSb \mid bSa \mid \epsilon$

Another complicated answer: $S \to aSb \mid bSa \mid aBa \mid bAb \mid \epsilon$

$B \to aBb \mid bBa \mid bSb$

$A \to aAb \mid bAa \mid aSa$

Parsed trees for $abba \in L_a$: $S \to SS \to aSbSa \to a\epsilon bbea \to abba$ (1st sol)

$S \to aBa \to abSba \to ab\epsilon ba \to abba$ (2nd sol)

(b) $L = \{w \in \{a,b\}^* \mid n_a(w) > n_b(w)\}$

$S \to Aa \mid MS \mid SMA$

$A \to Aa \mid \epsilon$

$M \to \epsilon \mid MM \mid bMa \mid aMb$

Parse trees for $baaba \in L_>$: $S \to MS \to MAa \to Ma \to MMa \to bMaa Mba$

$\to b\epsilon aa\epsilon ba \to baaba$

(c) $L_\neq = \{w \in \{a,b\}^* \mid n_a(w) \neq n_b(w)\}$

$L \to L_a \mid L_b$

$L_a \to L_= aL_a \mid L_= aL_=$

$L_b \to L_= bL_b \mid L_= bL_=$

$L_= \to aL_= bL_= \mid bL_= aL_= \mid \epsilon$

Parse trees for $bbaba \in L_\neq$: $L \to L_b \to L_= bL_= \to \epsilon bL_= \to bbL_= aL_=$

$\to bb\epsilon abL_= aL_= \to bbab\epsilon a\epsilon \to bbaba$

3. For a string $w \in \{a,b,c\}^*$, let us denote by $n_a(w)$ the number of a's in w, by $n_b(w)$ the number of b's in w, and by $n_c(w)$ the number of c's in w.

   (a) Design a general (unrestricted) grammar for the language

   $$L_{abc} = \{w \in \{a,b\}^* \mid n_a(w) = n_b(w) = n_c(w)\}.$$

   Give a derivation in your grammar for the string $cabacb \in L_{abc}$.

   (b) Prove (precisely!) that the language L_{abc} is not context-free. 15 points

a) Design the general unrestricted grammar

Wednesday, 30 March 2022 10.21

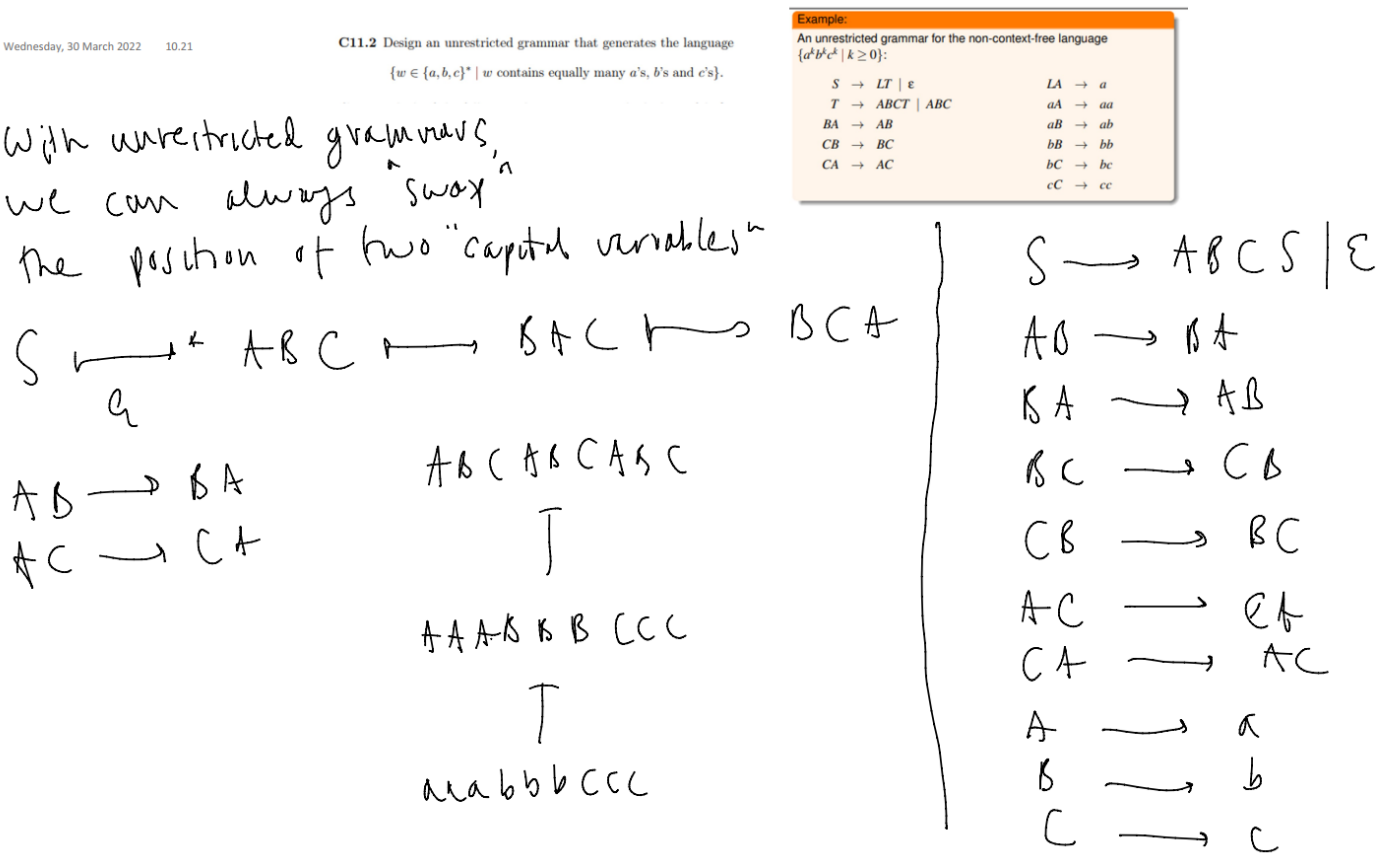
C11.2 Design an unrestricted grammar that generates the language

$\{w \in \{a,b,c\}^* \mid w \text{ contains equally many } a\text{'s, } b\text{'s and } c\text{'s}\}$.

Example:

An unrestricted grammar for the non-context-free language $\{a^k b^k c^k \mid k \geq 0\}$:

S	$\rightarrow$	$LT \mid \varepsilon$	
T	$\rightarrow$	$ABCT \mid ABC$	
BA	$\rightarrow$	AB	
CB	$\rightarrow$	BC	
CA	$\rightarrow$	AC	
LA	$\rightarrow$	a	
aA	$\rightarrow$	aa	
aB	$\rightarrow$	ab	
bB	$\rightarrow$	bb	
bC	$\rightarrow$	bc	
cC	$\rightarrow$	cc	



b) prove that the language in b is not context-free

Theorem: For any regular language L_0 and CFL L_1, language $L_0 \cap L_1$ must be a CFL.

Use this theorem, and the fact that $\{a^i b^i c^i \mid i \geq 0\}$ is not context-free, to show that the following language is not context-free:

The set of all strings over $\{a, b, c\}$ with an equal number of a's, b's and c's.

3. Let L_0 be the set of all strings over $\{a, b, c\}$ with an equal number of a's, b's and c's. **Assume that L_0 is context-free**. Let $L_1 = a^* b^* c^*$. Clearly L_1 is regular. But $L_0 \cap L_1 = \{a^i b^i c^i \mid i \geq 0\}$, and by the theorem it must be context-free. But we already showed that $\{a^i b^i c^i \mid i \geq 0\}$ is not context-free. So, **by contradiction, L_0 can not be a CFL.**

4. Which of the following claims are true and which are false? Provide a brief justification for each of your answers, based on results introduced at the course. (For example if the claim was: "The complement of any decidable language is semidecidable", your answer could be: "True. The complement of any decidable language is decidable (by switching the accepting and rejecting states in the recognising TM), and all decidable languages are by definition also semidecidable.")

   (a) The complement of any regular language is context-free.

   (b) The intersection of any two context-free languages is regular.

   (c) The union of any two context-free languages is context-free.

   (d) Nondeterministic Turing machines can recognise some undecidable languages.

   (e) The intersection of any two semidecidable languages is decidable. 15 points

(a) The complement of any regular language is context-free
False. The compkement of any regular language is also regular. Class of regular languages is closed under complement by switching accepting and non accepting states
(b) The intersection of any two context-free languages is regular
False. Intersection of languages in the chomsky hierachy can only be recognized from the highest level. For example, intersection of type 1 and type 2 requires type 2 machine to recognize the intersection. Now, CFG is type 1, and their intersection would also be a CFL as well, which is not a regular language
(c) The union of any two context-free languages is context-free
True. The class of CFG is closed under union, concatenation and star operations
(d) Nondeterministic Turing machines can recognize some undecidable languages
False. A language that the nondeterministic TM recognizes is called Turing-recognizable or semi-decidable or recursively enumerable languages. There is no automata that can recognize undecidable languages, such as halting and non emptiness problem.
(e) The intersection of any two semidecidable languages is decidable
False. The intersection and union of any two semi-decidable languages is also semi-decidable

C11.1 Prove that the following problems are undecidable:

(i) Given a Turing machine M, does M recognise a regular language?

(ii) Given a Turing machine M, does M accept exactly one input string?

(iii) Given Turing machines M_1 and M_2, is $\mathcal{L}(M_1) \cap \mathcal{L}(M_2) = \emptyset$?

(i) Given a Turing machine M, does M recognise a regular language?
(ii) Given a Turing machine M, does M accept exactly one input string?
(iii) Given Turing machines M1 and M2, is L(M1) intersects L(M) = 0?
Use Rice Theorem to prove all of them

C11.3 Which of the following claims are true and which are false?

(i) The computation of a deterministic Turing machine halts on every input.

(ii) The complement of any decidable language is semidecidable.

(iii) The intersection of any two semidecidable languages is decidable.

(iv) The problem of determining if a Turing machine accepts at least 7 strings is undecidable.

(v) The problem of determining if a Turing machine has at least 7 states is undecidable.

(vi) The problem of determining if a Turing machine runs for at least 7 steps on all inputs of length $|x| \leq 7$ is undecidable.

(i) The computation of a deterministic Turing machine halts on every input
True, since the Turing machine is deterministic, it has finite states and thus it will always halt
(ii) The complement of any decidable language is semidecidable.
False. The complement of decidable languages is also decidable. Decidable langs are closed under complement
(iii) The intersection of any two semidecidable languages is decidable.
False. The intersection union of any two semidecidable languages is also semi-decidable.
(iv) The problem of determining if a Turing machine accepts at least 7 strings is undecidable.
True, using Rice Theorem
(v) The problem of determining if a Turing machine has at least 7 states is undecidable.
False. The states of the Turing machine can be counted in a traditional fashion, and thus it is a decidable language
(vi) The problem of determining if a Turing machine runs for at least 7 steps on all inputs of length |x| <= 7 is undecidable.
True, using Rice Theorem

H9.3 Are the following statements true or false? Justify your answers.

1. All regular languages are decidable.
2. The complement of any context-free language is decidable.
3. The complement of any language recognised by a deterministic Turing machine is decidable.
4. All languages recognised by nondeterministic Turing machines are semi-decidable.
5. If a language can be recognised by a deterministic pushdown automaton, then it is also decidable.
6. If a language is decidable, then it can be recognised by some non-deterministic pushdown automaton.

1. All regular languages are decidable.
True. We know that the problem is decideable because a DFA for the regular language always quits after n steps on an input of size n, so the simulation is guaranteed to halt. The automaton always halt => regular languages are decidable

2. The complement of any context-free language is decidable:
The complement of a context-free language can be context-free or not; the complement of a non-context free language can be context-free or not => CFL is not closed under complement. Decidable languages are closed under complement, and context-free languages are decidable, so therefore the complement of a context-free language is also decidable.
3. The complement of any language recognised by a deterministic Turing machine is decidable.
Language is recognized by a deterministic Turing machine => it is decidable. Therefore, its complement will also be decidable as decidable languages are closed under complement

4. All languages recognised by nondeterministic Turing machines are semi-decidable.
True. Since it is recognized by a nondeterministic Turing machines, it will always halt on accepted string. But it is not sure to halt or loop forever for non-accepting strings => The accepted langs by nondeterminsitic Turing machines are semi-decidable
A semidecidable problem (or equivalently a recursively enumerable problem) could be:
- Decidable: If the problem and its complement are both semidecidable (or recursively enumerable), then the problem is decidable (recursive).
- Undecidable: If the problem is semidecidable and its complement is not semidecidable (that is, is not recursively enumerable).
Important note: a decidable (recursive) problem is also semidecidable (recursively enumerable). Conversely, if a problem is not recursively enumerable (semidecidable), then is not recursive (decidable).

5. If a language can be recognised by a deterministic pushdown automaton, then it is also decidable.
True, since PDA always halt after scanning the string => the language is decidable

6. If a language is decidable, then it can be recognised by some nondeterministic pushdown automaton.
True. If a language is decidable => it can be recognized by a deterministic Turing machine

Grammar	Languages	Automaton	Production rules (constraints)*	Examples[3]
Type-0	Recursively enumerable	Turing machine	$\gamma \to \alpha$ (no constraints)	$L = \{w \mid w$ describes a terminating Turing machine$\}$
Type-1	Context-sensitive	Linear-bounded non-deterministic Turing machine	$\alpha A \beta \to \alpha \gamma \beta$	$L = \{a^n b^n c^n \mid n > 0\}$
Type-2	Context-free	Non-deterministic pushdown automaton	$A \to \alpha$	$L = \{a^n b^n \mid n > 0\}$
Type-3	Regular	Finite state automaton	$A \to \mathrm{a}$ and $A \to \mathrm{a}B$	$L = \{a^n \mid n \geq 0\}$

* Meaning of symbols:

- a = terminal
- A, B = non-terminal
- α, β, γ = string of terminals and/or non-terminals
  - α, β = maybe empty
  - γ = never empty

Note that the set of grammars corresponding to recursive languages is not a member of this hierarchy; these would be properly between Type-0 and Type-1.

Every regular language is context-free, every context-free language is context-sensitive, every context-sensitive language is recursive and every recursive language is recursively enumerable. These are all proper inclusions, meaning that there exist recursively enumerable languages that are not context-sensitive, context-sensitive languages that are not context-free and context-free languages that are not regular.[4]