COT4210 Discrete Structures – Final Exam

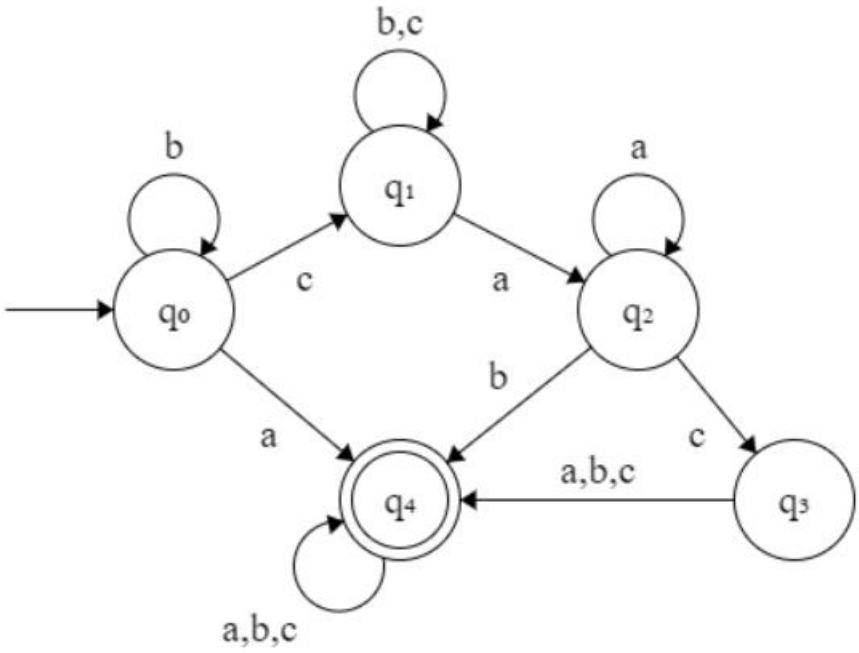
Spring 2023

Unit 1: Regular Languages

1. (15) Let *L* be the language over {**a**, **b**, **c**} accepting all strings **except** those so that:

    1. No **a**'s occur before the first **c**.
    2. No **b**'s occur after the first **a**.
    3. The last symbol of the string is **c**.
    4. No **a** is followed by a **c** unless that **c** is the last symbol.

    Choose any constructive method you wish, and demonstrate that *L* is regular. *You do not need an inductive proof, but you should explain how your construction accounts for each rule.*



    This valid DFA shows that the language L is regular. Since we want all strings except, we create a DFA that accepts all strings following the rules, then reverse the accept and non-accept states.

    q_0 and q_2 force the string to either break rule 1 or 2, and q_0, q_2 and q_3 make sure that if strings follow rule 3 and 4, they have to break rule 1 and 2.

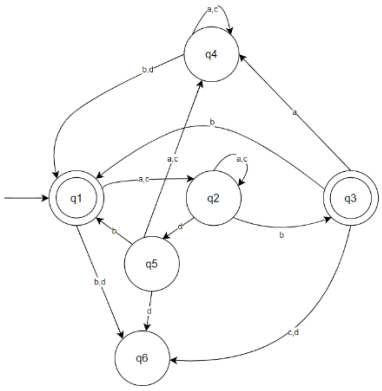
2. (15) Show using a cross-product construction that the class of regular languages is closed under intersection. *You do not need an inductive proof, but you should convincingly explain why your construction works.*

    Assume two languages L_1 and L_2 and their DFAs D_1 and D_2. We can create a DFA D where:

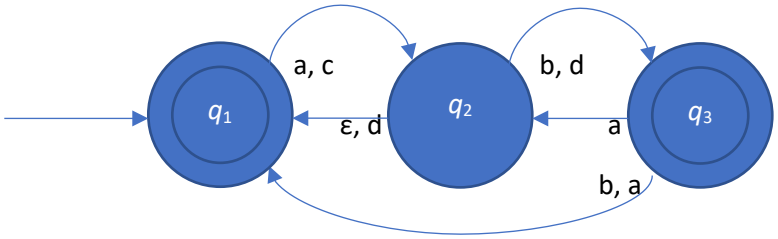
    1. $(Q_1 \times Q_2)$ is a finite set called the states
    2. Σ is a finite set called the alphabet
    3. $\delta : (Q_1 \times Q_2) \times \Sigma \longrightarrow Q_1 \times Q_2$ is the transition function
        Where $\delta\big((p_1, p_2), w\big) = (\delta_1(p_1, w), \delta_2(p_2, w))$ for all $p_1 \in Q_1, p_2 \in Q_2, w \in \Sigma^*$
    4. $(q_{0_1}, q_{0_2}) \in Q$ is the start state
    5. $(F_1 \times F_2) \subseteq Q$ is the set of accept states

Since we can create DFA D that represents the intersection of D_1 and D_2 for all languages, this proves that the class of regular languages is closed under intersection.

3. (20) Using the procedure demonstrated in class and in the textbook, convert this NFA to a DFA.



Answer->



Unit 2: Non-Regular Languages

4. (10) Let *L* = { *ccc#cc#c* with *c* ∈ {**a**, **b**}* }. Show that *L* is not regular.

Assume the complement language is regular, with $pumping\ length\ =\ p$. Consider a pumping string $s = ccc^p\#cc\#c$. So, we have $s = xyz$ with x the prefix, y the cycle string and z the suffix, so that:

$PL1: xy^iz \in L\ for\ all\ i \in N\ or\ xy^*z \in L$

$PL2: |y| > 0\ PL3: |xy| \leq p$

By definition of $PL1$, $PL2$ and our pumping string s, all xyz divisions of s must be:

- $x = c^a$
- $y = c^b$, where b must be greater than 0
- $z = c^p\#cc\#c$,
  - Since $|xy| \leq p$, the first two c's must be made of x and y, then the final c being part of c.
- So, the resulting string xyz should be equal to $c^ac^bc^p\#cc\#c$
- by $PL1$, set $i = 2$ and consider $xy^iz = xyyz$
  - Since $xyyz = c^ac^bc^bc^p\#cc\#c = c^ac^{2b}c^p\#cc\#c$
  - And $c^ac^bc^p\#cc\#c \neq c^ac^{2b}c^bc^p\#cc\#c$ violates our language
    - Since the ccc part of ccc#cc#c needs to equal c times three and thus violates our language.
  - Then by definition of $L, xyyz \notin L, \rightarrow\leftarrow PL1$

Therefore, for every possible construction of $\boldsymbol{y}$, $\boldsymbol{xyyz} \notin \boldsymbol{L}$, →← PL1 and by contradiction, the language must be irregular.

5. (20) Let *L* be the language over {**a**, **b**, **c**} accepting all strings so that:

1. No **a**'s occur before the first **c**.
2. No **b**'s occur after the first **c**.
3. The last symbol of the string is **c**.
4. There are fewer **b**'s than **a**'s.

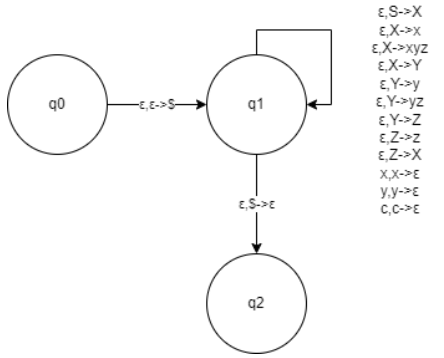
Construct a context-free grammar generating *L*. *You do not need an inductive proof, but you should explain how your construction accounts for each rule.*

$V = \{A, B, C\},$
$\Sigma = \{\boldsymbol{a}, \boldsymbol{b}\},$
$S \in A,$
$R = \{A \rightarrow BC, \quad B \rightarrow \boldsymbol{b}BC\boldsymbol{a}|C, \quad C \rightarrow \boldsymbol{c}D|\boldsymbol{ac}, \quad D \rightarrow \boldsymbol{a}D|\boldsymbol{c}D|\boldsymbol{a}\}$

6. (20) Let *G* = (*V*, ∑, *R*, *S*) be a grammar with *V* = {X, Y, Z}; ∑ = {**x**, **y**, **z**}; and the set of rules:

S → X
X → **x** | **xyz** | Y
Y → **y** | **yz** | Z
Z → **z** | X

a. (5) Convert *G* to a PDA using the method we described.



b. (15) Convert *G* to Chomsky normal form.

Add start S and remove ϵ:
$S_0 \rightarrow S|\epsilon$
$S \rightarrow X$
$X \rightarrow x|xyz|Y$
$Y \rightarrow y|yz|Z$
$Z \rightarrow z|X$

Remove single rewrites:
$S_0 \rightarrow XYZS_0|\epsilon$
$X \rightarrow x|X_0Y_0Z$
$Y \rightarrow y|Y_0Z$
$Z \rightarrow z$
$X_0 \rightarrow x$

$Y_0 \rightarrow y$

Remove mixed/multiple terminals:
$S_0 \rightarrow XYZS_0|\epsilon$
$X \rightarrow x|X_0Y_0Z$
$Y \rightarrow y|Y_0Z$
$Z \rightarrow z$
$X_0 \rightarrow x$
$Y_0 \rightarrow y$

Remove long rewrites to get CNF:
$S_0 \rightarrow X_1Z_0|\epsilon$
$X \rightarrow x|X_0Y_0Z$
$Y \rightarrow y|Y_0Z$
$Z \rightarrow z$
$X_0 \rightarrow x$
$Y_0 \rightarrow y$
$X_1 \rightarrow XY$
$Z_0 \rightarrow ZS_0$

This exam has two pages. Continue to the next page!

COT4210 Discrete Structures – Final Exam

Fall 2020 – Open Book, Open Notes –Good luck!

Unit 3: Computability and Complexity

7. (15) Let E_{DFA} = { <*D*> | *D* is a DFA and L(*D*) = ∅ } and ALL_{DFA} = { <*D*> | *D* is a DFA and L(*D*) = Σ* }. Show that both are in the class P.

$V =$ on input $< D >$:

Mimic E_{DFA} using D where:

- We run the DFA
  - While moving through each node
  - Reject if any node accepts
  - If we've searched the entire tree and there are no accept nodes then we accept.
    - This means that $L(D) = F$

Moving through a DFA's nodes is in polynomial time. Then E_{DFA} is in class P since it decides in polynomial time.

$V =$ on input $< D >$:

Mimic ALL_{DFA} using D where:

- We run the DFA
  - While moving through each node
  - If the node accepts, mark it and keep moving
  - If any non-accept states are found or we've searched the entire tree, reject.

Moving through a DFA's nodes is in polynomial time. Then ALL_{DFA} is in class P since it decides in polynomial time.

8. (15) Let $POWER_{TM}$ = { <*M*> | *M* is a TM, and for all *s* ∈ L(*M*), |*s*| is a power of 2 }. Show that $POWER_{TM}$ is undecidable. Do not use Rice's Theorem.

Proof: Assume BWOC that $POWER_{TM}$ is decidable

By definition of decidability let M_{POWER} be its decider where:

- We receive string s and $TM\ M$
- We accept if for all s ∈ L(M), |s| is a power of 2
- Otherwise reject

Then we build a M_{accept} where:

- We mimic A_{TM} and we receive Turing machine M that receives string s
- We construct a new Turing machine $M_{Internal}$ that,
  - Receives string x and
  - we reject if $|x|$ is not a string of infinite length, i.e a^*, ab^*, s^*
    - This insures that M always decides on items it fails to halt
  - Otherwise simulates M on x and mimics if M accepts, rejects or fails to halt

Interrogate $M_{Internal}$ for a match using M_{POWER} which causes M_{Accept} to decide A_{TM} and creates a contradiction. Therefore $POWER_{TM}$ must be undecidable.

9. (20) A graph *G* has an **independent set** of size *k* if there is a set *V* of *k* nodes in *G* so that no two nodes in *V* share an edge. Let *INDEPENDENT-SET* = { <*G*, *k*> | *G* is a graph with an independent set of size *k* }.

   a. (5) Show that *INDEPENDENT-SET* ∈ NP by writing either a verifier or an NDTM.

For INDEPENDENT-SET ∈ NP

We create a NDTM M' that accepts input G and k

- For input G and k
- Run M' on G and k
  - We can check if G of length k is an independent set by running through all the nodes, if there is a set V of k nodes where they do not share an edge then we can accept, otherwise reject.

    - This works well with a NDTM since we non-deterministically pick nodes until either we've tried all possible combinations of nodes, or we find an independent set. Either way it'll run in polynomial time
- Accepts when M' accepts.
- Rejects otherwise.

Then NDTM M' shows that INDEPENDENT-SET $\in NP$

b. (15) Show that *INDEPENDENT-SET* is NP-complete by reduction from *VERTEX-COVER*.