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.*

Diagram

Description automatically generated

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.

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

1. (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 and and their DFAs and . We can create a DFA D where:

1. is a finite set called the states

2. is a finite set called the alphabet

3. is the transition function

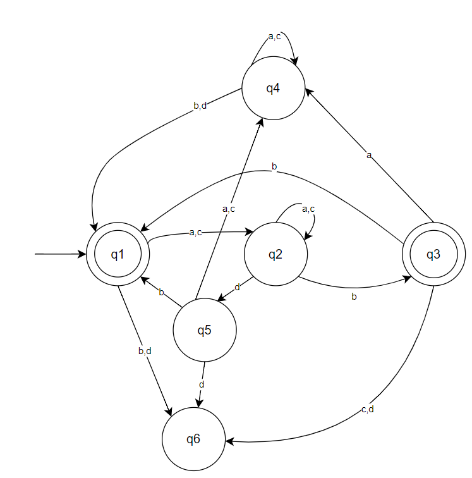
Where for all

4. is the start state

5. is the set of accept states

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

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

Answer->

a, c b, d

ε

, d

a

b, a



*q*

2

*q*

1

*q*

3

# Unit 2: Non-Regular Languages

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

Assume the complement language is regular, with . Consider a pumping string . So, we have 𝑠 = 𝑥𝑦𝑧 with 𝑥 the prefix, 𝑦 the cycle string and 𝑧 the suffix, so that:

By definition of , and our pumping string s, all divisions of s must be:

* , where b must be greater than 0
* ,
  + Since the first two ’s must be made of x and y, then the final c being part of c.
* So, the resulting string should be equal to
* by, set and consider
  + Since
  + And 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

Therefore, for every possible construction of 𝒚, 𝒙𝒚𝒚𝒛 ∉ 𝑳, →← PL1 and by contradiction, the language must be irregular.

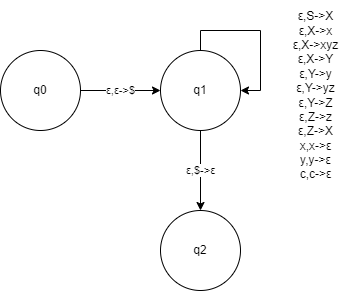
1. (20) Let *L* be the language over {**a**, **b**, **c**} accepting all strings so that:
   * 1. No **a**’s occur before the first **c**.
     2. No **b**’s occur after the first **c**.
     3. The last symbol of the string is **c**.
     4. There are fewer **b**’s than **a**’s.

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.*

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

S à X

* + - 1. à **x** | **xyz** | Y
      2. à **y** | **yz** |Z
      3. à **z** | X
  1. (5) Convert *G* to a PDA using the method we described.



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

Add start and remove :

Remove single rewrites:

Remove mixed/multiple terminals:

Remove long rewrites to get CNF:

**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

1. (15) Let *E*DFA = { <*D*> | *D* is a DFA and L(*D*) = F } and *ALL*DFA = { <*D*> | *D* is a DFA and L(*D*) = S\* }. Show that both are in the class P.

on input :

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

Moving through a DFA’s nodes is in polynomial time. Then *E*DFA is in class P since it decides in polynomial time.

on input :

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.

1. (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 is decidable

By definition of decidability let be its decider where:

* We receive string and
* We accept if for all s L(M), |s| is a power of 2
* Otherwise reject

Then we build a where:

* We mimic and we receive Turing machine M that receives string s
* We construct a new Turing machine that,
  + Receives string and
  + we reject if is not a string of infinite length, i.e
    - This insures that always decides on items it fails to halt
  + Otherwise simulates on and mimics if accepts, rejects or fails to halt

Interrogate for a match using which causes to decide and creates a contradiction. Therefore must be undecidable.

1. (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* }.
2. (5) Show that *INDEPENDENT-SET*  NP by writing either a verifier or an NDTM.

For INDEPENDENT-SET

We create a NDTM that accepts input G and k

* For input G and k
* Run 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 accepts.
* Rejects otherwise.

Then NDTM shows that INDEPENDENT-SET

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