CSCE 355
4/22/2024

Review day

①

Course outline:

- Math prelims: ~~inst~~ induction, pigeonhole principle

- Regular languages:
  - DFAs, NFAs ε-NFAs, regexes & conversions between them
  - DFA minimization
  - Simulating an NFA on an input string
  - Closure properties of the class of regular languages
  - Pumping Lemma for reg langs

- Context-free languages
  - Context-free grammars (CFGs)
  - Derivations (leftmost & rightmost)
  - Parse trees
  - Ambiguity
  - Push-down automata (PDAs): acceptance via
    - final state
    - empty stack
    - conversions
  - IDs of a PDA, comp paths
  - Restricted PDAs
  - CFG → PDA
  - PDA → restricted PDA → CFG

Pumping Lemma & closure properties for CFLs

- Turing machines (TMs)
  - definition
  - IDs & computations
  - Church-Turing thesis: TMs = Algorithms
  - Universal TM
  - Acceptance problem, A_{TM} is undecidable.
  - Editing problem (EP) is undecidable
  - ALL_{CFG} is undecidable
  - Intersection problem for CFLs is undecidable

variant of EP is possible

don't worry

Ex: Pumping Lemma for reg langs.

$L := \{ x \in \{0,1\}^* : x$ has a 0 somewhere in its 2nd half (not incl. middle digit if $|x|$ is odd$\}$

Prop: L is not reg lang. pumpable.

Proof: Given $p > 0$, let $s := \underline{0^p 0 1^{p-1}}$

$s \in L$ and $|s| \geq p$.

Given x, y, z such that

1) $s = xyz$
2) $|xy| \leq p$
3) $|y| > 0$,

Let $i :=$ 0

Then $xy^iz \notin L$, because ... [justify/explain]

$\therefore$ L not pumpable //

Have $y = 0^k$ for some $k \geq$

Thus $xy^0z = xz = 0^{p-k}01^{p-1} = 0^{p-k+1}1^{p-1} \notin L$

Ex: Closure properties: L any language

Def: DROP-ONE(L) is the set of strings formed from nonempty strings in L by removing a single character.

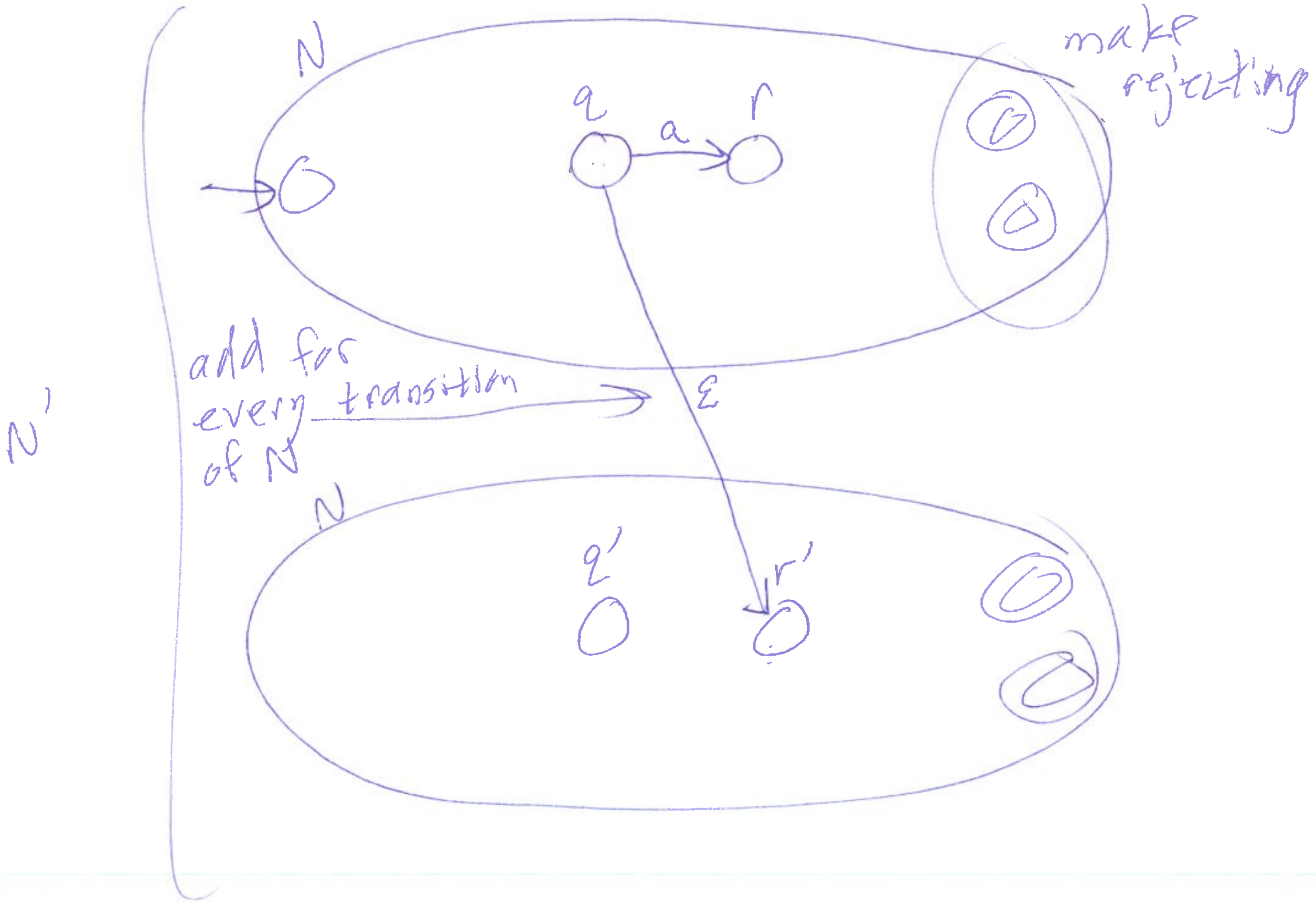
$$\text{DROP-ONE}(L) = \{xy : x, y \in \Sigma^* \text{ and } xay \in L \text{ for some } a \in \Sigma\}$$

$$\text{DROP-ONE}(\{abca, a, \varepsilon\})$$

$$= \{bca, aca, aba, abc, \varepsilon\}$$

Prop: If L is regular, then DROP-ONE(L) is regular.

Proof: Method 1: Given an ε-NFA N for L, convert to an ε-NFA N' for DROP-ONE(L):



Method #2: regex r for L into a regex r' for DROP-ONE(L):

r	r'
$\emptyset$	$\emptyset$
$a \in \Sigma$	$\varepsilon \ (= (\emptyset^*))$
$s+t$	$s' + t'$
st	$s't + st'$
s^*	$s^* s' s^*$

I won't ask you to construct a TM formally.

I may give you a TM and

- ask for a formal computation (sequence of IDs) on a given input
- describe the language decided/recognized by the given TM
- describe what the TM does

PDA to CFG

$\Gamma = \{Z_0, +\}$

✓ '(', Z_0 / push +

'(', + / push +

✓ ')', + / pop

✓ ε, Z_0 / pop

Grammar

Vars: S, $[qZ_0q]$, $[q+q]$

$S \rightarrow [qZ_0q]$

$[qZ_0q] \rightarrow \varepsilon$

$[q+q] \rightarrow$ ')'

$[qZ_0q] \rightarrow$ '(' $[q+q][qZ_0q]$

$[q+q] \rightarrow$ '(' $[q+q][q+q]$

$[qXr]$

Universal TM:

U := "On input $M\#w$, where M is a TM and w is ~~an~~ a string in M's input alphabet:

1. Simulate M on input w (and do what M does in terms of accepting/rejecting/outputting)"