CSCE 355
2/5/2025

Regular Expressions

Patterns that match strings

Fix an alphabet Σ.

~~A re~~ Def. A regular expression (regex) ~~is~~ over Σ is built from the following syntax:

Atomic: $\emptyset$, a $\left(\text{for } a \in \Sigma\right)$

$\emptyset$ — does not match anything

a — matches the string "a" and nothing else

Nonatomic: Let s and t be regexes over Σ.

Then

(union) $s+t$ — string can match s or t (or both)

(concat.) st — matches concatenations of strings matching s followed by strings matching t.

(Kleene closure, star operator) s^* — matches the concatenation of any number (zero or more) of strings matching s.

are regexes

$+$ & concat are binary ops (infix)

$*$ is unary postfix

precedences (lowest to highest): $+$, concat, $*$

Can use parens to control grouping.

Def: Every regex r over Σ defines a language $L(r) \subseteq \Sigma^*$. {The strings that match r, or ~~the~~ equiv, that r matches.}

Formally, $L(r)$ is defined inductively on the syntax of r:

$$L(\emptyset) = \emptyset$$

$$(a \in \Sigma)\quad L(a) = \{a\}$$

s, t regexes

$$L(s+t) = L(s) \cup L(t)$$

$$L(st) = L(s)L(t)$$

[**Def:** $L_1, L_2 \subseteq \Sigma^*$: $L_1 L_2 := \{xy : x \in L_1 \ \&\ y \in L_2\}$

$$L(s^*) = \underbrace{\{\varepsilon\}}_{L(s)^0} \cup \underbrace{L(s)}_{L(s)^1} \cup \underbrace{L(s)L(s)}_{L(s)^2} \cup L(s)L(s)L(s) \cup \cdots$$

$$\cdots \cup L(s)^n \cup \cdots$$

Ex: $\Sigma = \{a, b\}$

$$a + bb = a + (bb)$$

$$L(a+bb) = L(a+(bb)) = L(a) \cup L(b)L(b)$$

$= \{a\} \cup \{b\}\{b\} = \{a\} \cup \{bb\}$

$= \{a, bb\}$

$L((a+b)b) = L(a+b)L(b) = (L(a) \cup L(b))L(b)$

$= (\{a\} \cup \{b\})\{b\} = \{a, b\}\{b\}$

$= \{ab, bb\} \neq L(a+bb)$

$L(a^*) = \{\varepsilon, a, aa, aaa, \ldots, a^n, \ldots\}$

$L((ab)^*) = \{\varepsilon, ab, abab, ababab, \ldots, (ab)^n, \ldots\}$

$L(ab) = \{ab\}$

$L((a+b)^*) = \{\varepsilon, a, b, aa, ab, ba, bb,$

$aaa, aab, aba, abb, baa, \ldots\}$

all possible strings over $\Sigma = \{a, b\}$

$\{ \Sigma^* = $ set of all strings over $\Sigma \}$

$\Sigma^* = \{a, b\}^*$

$L(\emptyset^*) = \{\varepsilon\} \cup \emptyset \cup \underbrace{\emptyset\emptyset}_{=\emptyset} \cup \cdots \cup \cdots$

$= \{\varepsilon\}$

Let ε ~~be the r~~ be the regex $\emptyset^*$

Then $L(\varepsilon) = \{\varepsilon\}$. ("syntactic sugar")

More syntactic sugar:

Let r, s be regexes.

$r|s$ means $r + s$

$r+$ means $rr^* = r(r^*)$

"one or more occurrences of r"

$r?$ means $r \cup \varepsilon$

"optional r, or zero or one occurrence of r"

~~"abc"~~

"abc" means abc

"a+b" means "a" "+" "b"

(+ treated as an alphabet symbol)

• (period) means Σ (matches any single char from the alphabet)

If $\Sigma = \{a, b, c\}$, then $L(\bullet) = \{a, b, c\}$

Character classes

[abc] = a + b + c, lang is {a, b, c}

square brackets

[abc] = [bca] = ...

Σ = ASCII char set

[a-z] = [abcd...xyz] lowercase letters

[A-Z] = [AB...XYZ] capital letters

[^a-z] — matches anything except a...z

complement

Regexes for common prog. lang. constructs:

— unsigned integer constants

[0-9]+

— identifiers

[A-Za-z_][A-Za-z0-9_]*

— optionally signed int constants

$$[+-]?[0-9]+$$

— (unsigned float constants) a la Pascal:

one or more digits
then decimal point
then one or more digits
then option exponent following E or e

$$[0-9]+"."[0-9]+([Ee][+-]?[0-9]+)?$$ (exponent)

Theorem: For every regex r over Σ, there is an ε-NFA N over Σ such that $L(r) = L(N)$ (r & N are equivalent)

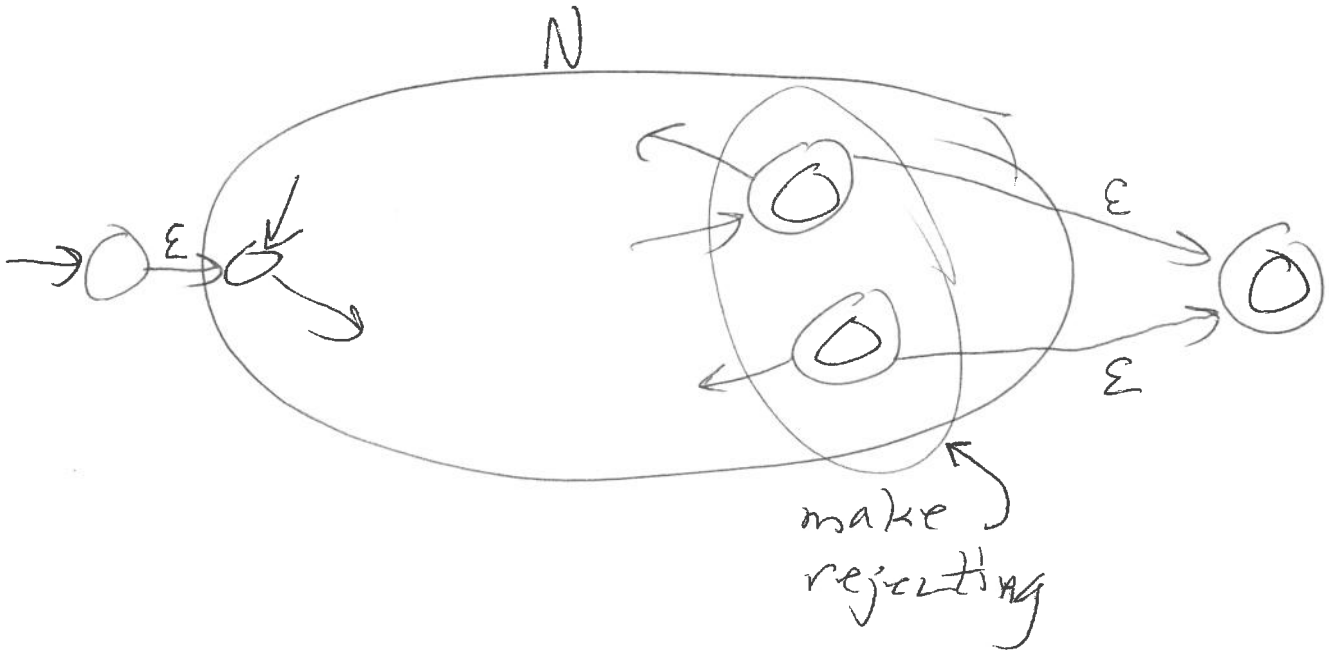
Proof is by construction.

Def: An ε-NFA is clean if
— it has exactly one accept state, and this state is distinct from the start state

- there are no transitions into the start state
- there are no transitions out of the accepting state.

Prop': For every Σ-NFA there is an equiv clean NFA.

Proof: Let N be any Σ-NFA



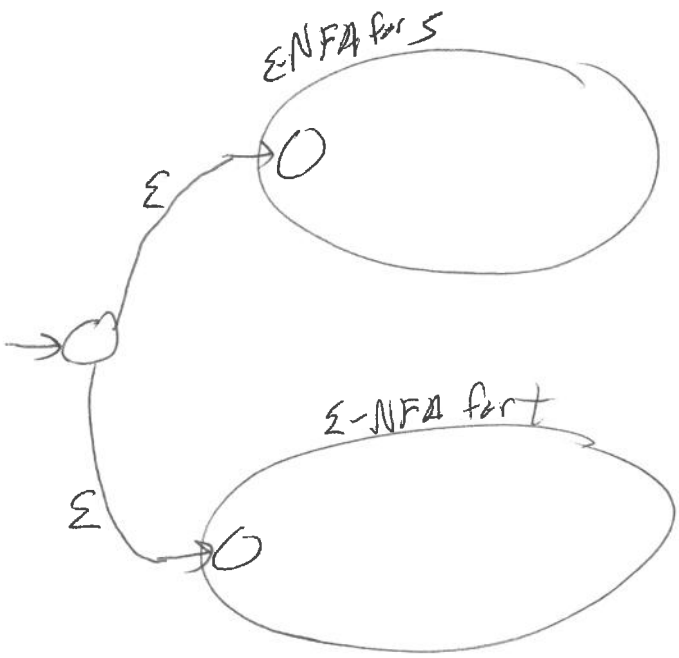
Correctness proof omitted

Proof of the thm (the construction) by induction on the regex syntax:

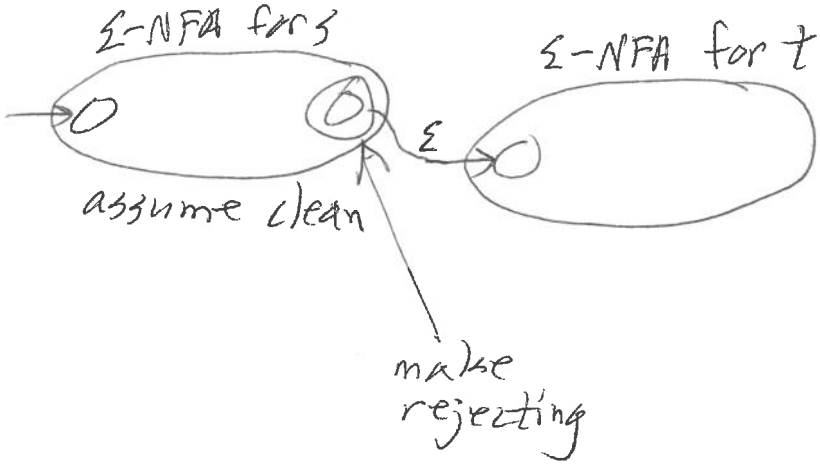
$\emptyset$

$(a \in \Sigma)$ a

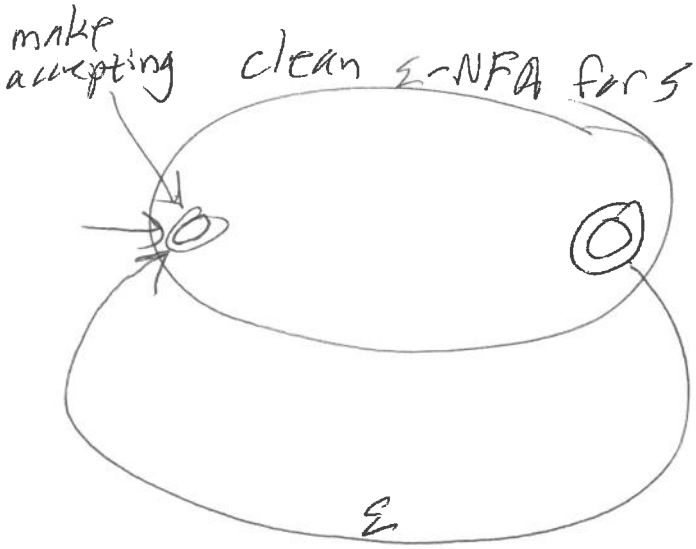
$s + t$


ε-NFA for s
ε
ε-NFA for t
ε

st


ε-NFA for s
ε-NFA for t
ε
assume clean
make rejecting

s^*


make accepting
clean ε-NFA for s
ε