CSCE 355
2/4/2026

Regular Expressions (regexes)

Time permitting: DFA min example

Def. Let A & B be languages over Σ.

$AB := \{xy : x \in A \text{ & } y \in B\}$ (concat of A & B)

$A^* := \{\varepsilon\} \cup A \cup AA \cup AAA \cup \dots \cup A^n \cup \dots$

$A^0 \quad A^1 \quad A^2 \quad A^3$

$$= \bigcup_{n \geq 0} A^n$$

Concat, $*$-operator ("Kleene closure"), & union ($A \cup B$) are the regular operators. A regular expression (regex) is an expression that represents (denotes) a language built from simple languages using regular operators.

Given a regex r, we let $L(r)$ be the lang denoted by r.

Define regex syntax & semantics recursively as follows

	regex r	$L(r)$
atomic regexes	$\emptyset$	$\emptyset$ (empty lang)
atomic regexes	a	$\{a\}$
$\forall a \in \Sigma$		

non-atomic regexes		
~~for regexes s, t~~		
(s,t regexes)	$s + t$	$L(s) \cup L(t)$
	st	$L(s)L(t)$
	s^*	$L(s)^*$

Parens can be used for grouping to override precedence rules:

$+$ — lowest prec.

concat — next higher

$*$ — highest

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

ab^* $\quad L(ab) = L(a)L(b) = \{a\}\{b\} = \{ab\}$

$abacca$ $\quad L(abacca) = \cdots = \{abacca\}$

$(a+b)(c+a)$ $\quad L((a+b)(a+c)) = L(aa + ac + ba + bc)$

$= \{aa, ac, ba, bc\}$

$+$ is commutative: $\quad r + s \equiv s + r$

$\equiv$ ← equivalent; denoting the same lang

assoc $\quad (r + s) + t \equiv r + (s + t)$

concat distributes over $+$:

$r(s+t) \equiv rs + rt$

$(s+t)r \equiv sr + tr$

concat is not commutative (order matters), but
" is associative (grouping does not matter)

a^* $L(a^*) = \{a\}^* = \{\varepsilon\} \cup \{a\} \cup \{a\}\{a\} \cup \cdots$

$= \{\varepsilon, a, a^2, a^3, \ldots, a^n, \ldots\}$

$= \{a^n : n \geq 0\}$

ab^* $L(ab^*) = \{a\}\{b\}^* = \cdots = \{a, ab, abb, \ldots, ab^n, \ldots\}$

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

$(a+b)^*$ $L((a+b)^*)$

$= \{\varepsilon\} \cup \{a, b\} \cup \{a, b\}\{a, b\} \cup \cdots$

$= \{\varepsilon, a, b, aa, ab, ba, bb, aaa, aab, \ldots\}$

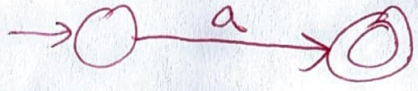
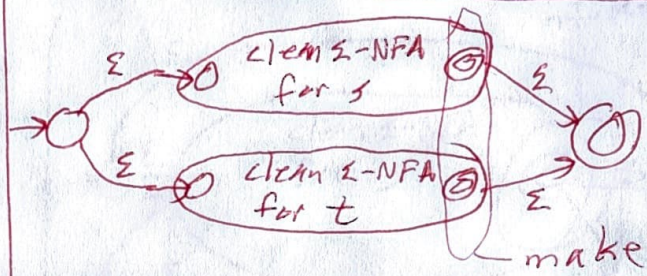
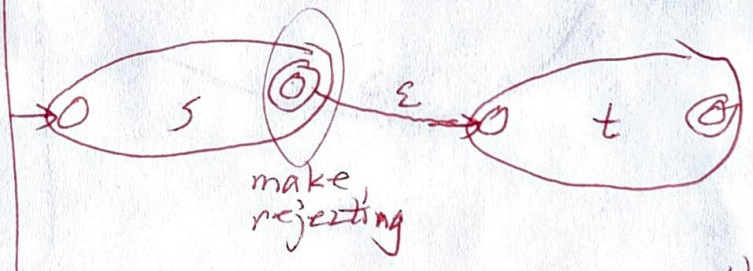
all strings over $\{a, b\}$

(Σ^* is just $*$-operator on Σ)

$a + b^*$ $L(a + b^*) = \{a, \varepsilon, b, b^2, b^3, \ldots\}$

Theorem: $L(r)$ is regular for every regex r.

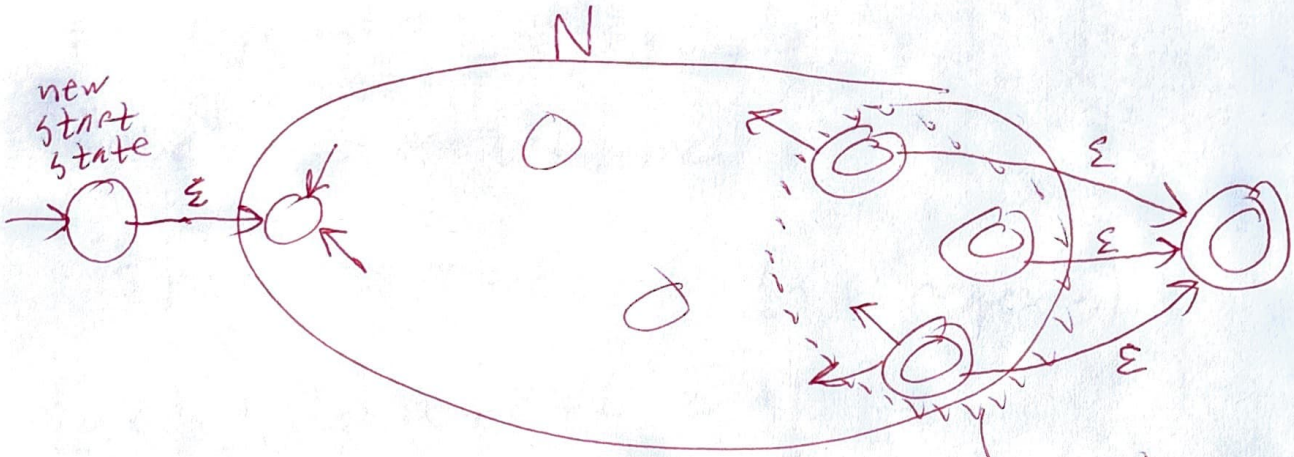
Proof: by construction. Given regex r we construct a equivalent ε-NFA N such that $L(N) = L(r)$.
($\because \exists$ DFA equiv to r by previous results).
Construction is recursive on syntax of r:

r	equivalent ε-NFA (clean)
$\emptyset$	
$(a \in \Sigma)$ a	
(s,t regexes) $s + t$	
st	

Def: An ε-NFA is clean if (a) It has exactly one accepting state, different from the start state, (b) No transitions out of the accepting state; (c) no transitions into the start state.

Prop: Every ε-NFA has an equivalent clean NFA.

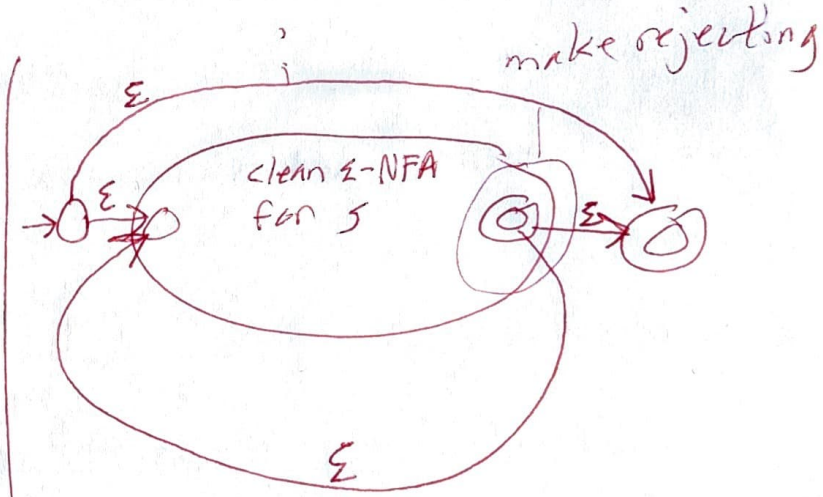
Proof: Given an ε-NFA N:



Construction (continued)

⋮

S^*



Regex shorthand (in popular use)

r regex

$r^+ :\equiv rr^*$ (one or more r's)

$\epsilon :\equiv \emptyset^*$

$L(\epsilon) = L(\emptyset^*) = \{\epsilon\} \cup \emptyset \cup \emptyset\emptyset \cup \emptyset\emptyset\emptyset \cup \cdots = \{\epsilon\}$

r? (optional r) $:\equiv r + \varepsilon$

"abc" $:\equiv$ abc

$s|t :\equiv s + t$ (s, t regexes)

$[a<b] :\equiv a + c + b \equiv a|b|c$

$[a-z] :\equiv a+b+c+\cdots+z$

[0-9] — any digit

[A-Za-z0-9] — any alphanumeric character

. — any char (except \n)

[^...] — any single char other than what is ...

Unsigned integer constants [0-9]+ (one or more digits)

Identifiers in C, C++, Java [A-Za-z_][A-Za-z0-9_]*

Unsigned Floating point constants (in Pascal)

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

Let r be a regex and w a string.

"r matches w"
"w matches r" } just mean $w \in L(r)$