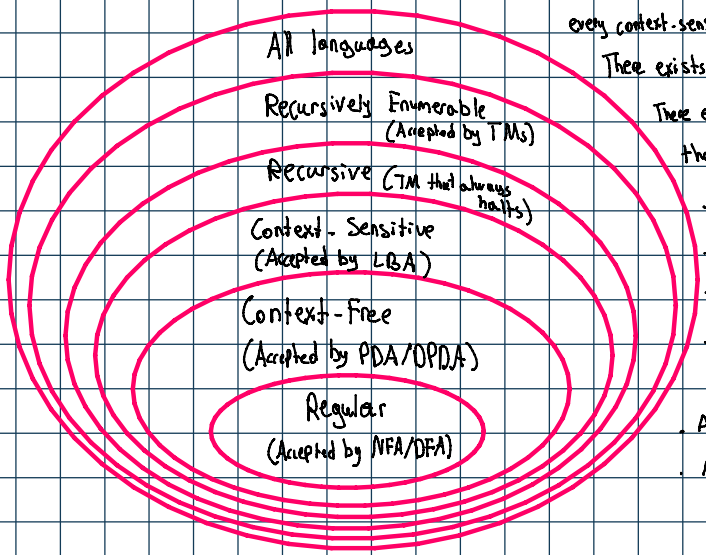
Cheat sheet

December 9, 2020 3:38 PM

* Closure properties of different classes of languages

Operation	Regular	Deterministic context-Free	context-Free	context-Sensitive	Recursive	Recursively enumerable
Union	yes	NO	yes	yes	yes	yes
Intersection	yes	NO	NO	yes	yes	yes
Set Difference	yes	NO	NO	yes	yes	NO
Complement	yes	yes	NO	yes	yes	NO
Concatenation	yes	NO	yes	yes	yes	yes
Kleene star	yes	NO	yes	yes	yes	yes
Kleene plus	yes	NO	yes	yes	yes	yes
Reversal	yes	yes	yes	yes	yes	yes
Homomorphism	yes	NO	yes	NO	NO	yes
Inverse Homomorphism	yes	yes	yes	yes	yes	yes
λ-free Homomorphism	yes	NO	yes	yes	yes	yes
Intersection with Regular language	yes	yes	yes	yes	yes	yes

NO in this table is for uncertainty. For example if L_1 and L_2 are context-Free then $L_1 \cap L_2$ is not necessarily context-Free.



- Every regular language is context-Free, every context-free language is context-sensitive, every context-sensitive language is recursive, every recursive language is recursively enumerable. There exists recursively enumerable languages that are not context-sensitive.
- There exists context-sensitive languages that aren't context-free. And context-Free languages that are not regular.
- All finite languages are regular
- The class of DFAs is equivalent to the class of NFAs,
- Every NFA can be converted to DFA. DFA is already also an NFA.
- The class of (N)PDA is not equivalent to DPDA. Some languages accepted by (N)PDA cannot be accepted by a DPDA.
- All the TM classes have the same power (not efficiency) as the standard TM.
- Multi-track is not a different variation cuz it just a larger tape still 1 head.

* Some example languages and what class they fit in.

① $L = \{ww \mid w \in (a+b)^*\}$ is context-Sensitive

→ we can't use NFA for this. If we use a PDA, once w is in the stack we can only read it in reverse so we would need 2 stacks. This language can be accepted by a Turing machine in linear space. So its context-sensitive.

② $L = \{ww \mid w \in (a+b)^+\}$ is context-Sensitive.

→ we can't use NFA for this. If we use a PDA, once w is in the stack we can only read it in reverse so we would need 2 stacks. This language can be accepted by a Turing machine in linear space. So its context-sensitive.

③ $L = \{ww^R \mid w \in (a+b)^*\}$ is context-Free

→ ww^R can be accepted by a PDA, but we would need an NPDA, cuz there is no deterministic way for us to tell when w ends and

③ $L = \{ ww^R \mid w \in (a+b)^* \}$ is Context-Free

→ ww^R can be accepted by a PDA, but we would need an NPDA, cuz there is no deterministic way for us to tell when w ends and where w^R begins. So it's a Context-Free language. If we want it to be deterministic, accepted by a DPDA, it would have to be like the exercise on the assignment so $L = \{ wcw^R \mid w \in (a+b)^* \}$. This is a Deterministic context-Free language.

④ $L = \{ www^R \mid w \in (a+b)^+ \}$ is Context-Free

→ Same shit as previous.

⑤ $L = \{ wxw \mid w, x \in (a+b)^* \}$ is Regular

→ we can set w to be λ, then the language is just $x = (a+b)^*$ so $\{ \lambda, a, b, aa, ab, ba, bb... \}$ which is Σ^*

Since we just expressed this language as a regular expression. its regular kek.

⑥ $L = \{ wxw \mid w, x \in (a+b)^+ \}$ is Context-Sensitive

→ This language includes all strings starting and ending with a, or starting and ending with b. It also includes strings that start with a and end with b. But abab isn't included, abbab and aabbbabaab are, so we need the string to have matching substrings for prefix and suffix with at least one letter in between. A machine to accept this language is a Turing Machine with linear space so its context sensitive.

⑦ $L = \{ wxw^R \mid w, x \in (a+b)^* \}$ is Regular

→ we can set w to be λ, then the language is just $x = (a+b)^*$ so $\{ \lambda, a, b, aa, ab, ba, bb... \}$ which is Σ^*

Since we just expressed this language as a regular expression. its regular kek.

⑧ $L = \{ wxw^R \mid w, x \in (a+b)^+ \}$ is Regular

→ This language contains all strings starting and ending with a or starting and ending with b, also its needs to be at least 3 letters.

Example accepted strings $\{ aaa, aba, aaaa, aaba, abaa, abba, baab... \}$ This can be accepted by an FSM or represented by $(a(a+b)a+b(a+b)b)(a+b)$ so its regular.

⑨ $L = \{ wxwy \mid w, x, y \in (a+b)^+ \}$ is Regular

→ This language contains all strings that have the substring/character w repeat between the second and the last character x and y. w can just be a or b to keep it short. we can express this language as $a(a+b)a(a+b)+b(a+b)b(a+b)$ so its regular.

⑩ $L = \{ xwyw \mid x, w, y \in (a+b)^+ \}$ is Regular

→ This is the same logic as previous one except we switch it with w being last character so $(a+b)a(a+b)a+(a+b)b(a+b)b$ so its regular.

⑪ $L = \{ wxyw \mid x, w, y \in (a+b)^+ \}$ is Context-Sensitive

→ In this language we can't apply the same logic of fixing w to be a or b because strings can start with a and with b and still be accepted. For example if $w = ab$, $x = y = a$ then we get abaaab which is accepted in the language. We need to compare the substrings at the beginning and at the end so it's a context-sensitive language.

⑫ $L = \{ xww \mid x, w \in (a+b)^* \}$ is Regular

→ In this language we can set w to λ so $x \in (a+b)^*$ making it regular with $L = \Sigma^*$ like ⑤ and ⑦.

⑬ $L = \{ xww \mid w, x \in (a+b)^+ \}$ is Context-sensitive

→ Here we can't have w being empty. we need an LBA to accept this language so it's context-sensitive.

⑭ $L = \{ xww^R \mid w, x \in (a+b)^+ \}$ is Context-Free

→ Again we can't have λ and we need to reverse the w, so we use the stack. So a PDA is needed for this making it context-free. Our PDA needs to guess the limits of w so we can't have Deterministic choices so its an NPDA.

* More examples with no proofs:

$L = \{ a^i b^j c^k \mid i, j, k > 0,\ i = j \text{ or } i = k \}$ Context-Free

$L = \{ a^i b^j c^k \mid i = k,\ j > 2 \}$ Context-Free

$L = \{ 0^n \mid n \text{ is a multiple of } 335 \}$ Regular

$L = \{ 0^i 1^j 2^k \mid i + j = k \}$ Context-Free

$L = \{ \langle M, w \rangle \mid w \in \Sigma^*,\ M = \text{DFA},\ M \text{ accepts } w \}$ Recursive

$L = \Sigma^*$ Regular

$L =$ some finite set Regular

$L = M_{TM}$ Halting Problem Recursively enumerable

$L = \{\} = \emptyset$ Regular

$L = \{a^{p+1} \mid p \text{ is prime}\}$ Recursive

$L = \{w \mid w \in \{a+b\}^*, w \text{ is a palindrome}\}$ Context-Free

$L = \{w \mid w \in \{a+b\}^*, n_a = n_b\}$ Context-Free

$L = \{a^i b^j a^k \mid i=j=k\}$ Recursive

* Formal Definitions Recap

. Deterministic Finite Automata . DFA

$M = (Q, \Sigma, \delta, q_0, F)$

Q = set of all states

Σ = set of symbols, Alphabet.

δ = Transition functions to map states, symbols to a new state. $Q \times \Sigma \to Q$

q_0 = initial state. $q_0 \in Q$

F = set of accepting/Final states. $F \subseteq Q$

. Non-Deterministic Finite Automata . NFA

$M = (Q, \Sigma, \delta, q_0, F)$

Q = set of all states

Σ = set of symbols, Alphabet.

δ = Transition functions to map states, symbols to a new state. $Q \times \Sigma \to Q$.

This variation of the function allows λ-transitions, have dead/trap states and "be" at multiple states at once.

q_0 = initial state. $q_0 \in Q$

F = set of accepting/Final states. $F \subseteq Q$

. Convert NFA to DFA

. Subset construction. NFA states are subset of DFA states. An NFA with n states, will result in a DFA with $\leq 2^n$ states

$Q' = 2^Q$, start with it empty. Each state represents a subset of states from M.

$\Sigma' = \Sigma$

δ' = For each NFA state, follow as a DFA transition of each element in the set making a new subset, add to Q' and apply λ-transitions

q_0' = initial state of new FSM/DFA.

F' = All DFA states that contain at least 1 original NFA final state.

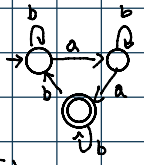
. How to Find number of states in a minimal DFA

. Draw the DFA, do the minimalization procedure with the table and equivalences to reduce it.

No shortcut or formula for this I guess :(

. How to count number of strings of length n accepted by a FSM

. Example from Quiz. How many strings of length 5 over $\{a,b\}$ are accepted by this NFA.



So here we have 2 types to count. 1) strings when we do a whole loop 2) no cycle

① only one string accepted aabaa by doing a full cycle.

② we count the number of strings of length 5 with two as. so 5 choose 2; $\binom{5}{2} = 10$.

① + ② => 1 + 10 = 11 strings

. The pumping lemma

. Given a regular language R. The pumping lemma says that if we have a long string (a string with length greater than the number of states in the FSM), there has to be a loop somewhere. So we can split it into 3 parts xyz, with y being the loop. Now fancier formal version.

- If R is a regular language then there exists an integer p where if s from R is any string of length at least p, then s may be divided into parts $s = xyz$, such that:
  - ① $\forall i \geq 0$, $xy^iz \in R$. If we pump the loop part 0 or more time, the resulting string still belongs to the language.
  - ② $|y| > 0$. The loop part has to exist, it should have at least one state.
  - ③ $|xy| \leq p$. The lead-up to the loop added with the loop itself need to fit in our pumping number. Basically the loop has to happen in first p chars

- The pumping lemma for context-free languages.