One more way of constructing new regular languages from old ones:

9/17/18

I have 2 regular languages (both on Σ) L_1 & L_2 with DFAs M_1 & M_2.

Define

$$L_1/L_2 = \{ x \in \Sigma^* \mid \text{there exists } y \in L_2 \text{ so that } xy \in L_1 \}$$

stupid case: if $L_2 = \{\lambda\}$, then

$$L_1/L_2 = L_1$$

Suppose

$$L_1 = \{abaab, abab\}, \quad L_2 = \{ab, bab\}.$$

Then

$$L_1/L_2 = \{a, ab, aba\}$$

$L_1 = L(a^* b^*)$

$L_2 = L(b^*)$

$L_1 / L_2 = L(a^* b^*)$

$L_1 = a^* b^*$

$L_2 = a b^*$

Then $L_1 / L_2 = a^*$

Why is L_1 / L_2 regular?

I have some string x, and automata M_1 & M_2. How do I tell if $x \in L_1 / L_2$?

① Run x on M_1 - I will get to a state q. I want to know if there is some string y that

a) is accepted by M_1 starting at q

b) is accepted by M_2 (starting at its start state)

Back up for a moment:

I can tell if a DFA accepts anything at all by seeing if there is a final state reachable from the start state.

Back up a little less

To tell if there is a string accepted by both of 2 machines:

Construct the machine "$M_1 \times M_2$" (~~that we~~ like we did last Friday to accept $L_1 \cap L_2$) and see if it has a final state reachable from its start state.

Return to original problem:

Take M_1, modified so start state is q, and M_2 - construct the product machine, and see if it has any ~~sta~~ final states reachable from start state.

This answer works for any ~~thing~~ x ending at q.

If I want an answer for every string, I have to do this computation for every state of M_1.

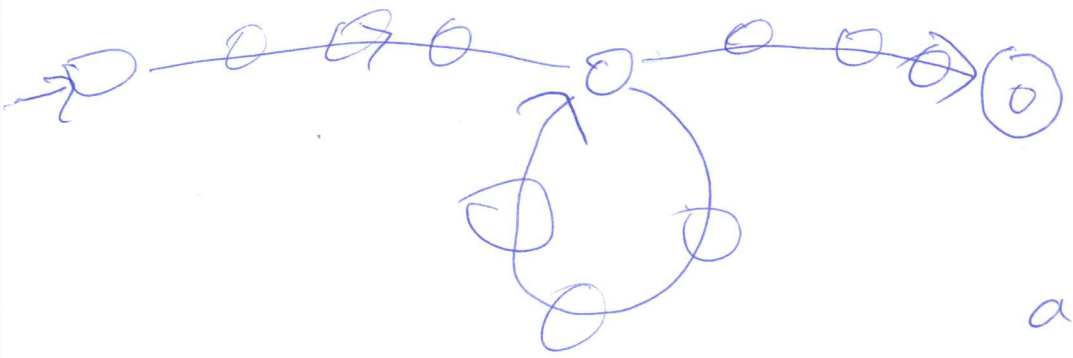
My machine accepting L_1/L_2 looks like M_1, except that my final states are exactly the states for which this answer is "Yes."

Things we could write a computer program to tell about a given regular language. (specified ~~as~~ e.g. as a DFA)

a) if there is any string in the language at all.

b) if the language has infinitely many strings

check if we have a loop that



1) is reachable from the initial state

and

2) has a path to a final state.

or - check if a reg. expr. for it has any *'s.

c) check if two ~~machines~~ DFAs M_1 & M_2 accept the same language.

Can we check if there are any strings accepted by M_1 ~~but not M_2~~ and rejected by M_2? We can flip final/non-final states of M_2 - this gives a machine $\overline{M_2}$ that accepts precisely the strings rejected by M_2.

Any strings accepted by both M_1 & $\overline{M_2}$?

Yes - construct the "product machine" $M_1 \times \overline{M_2}$ - see if it has any final states reachable from the initial state.

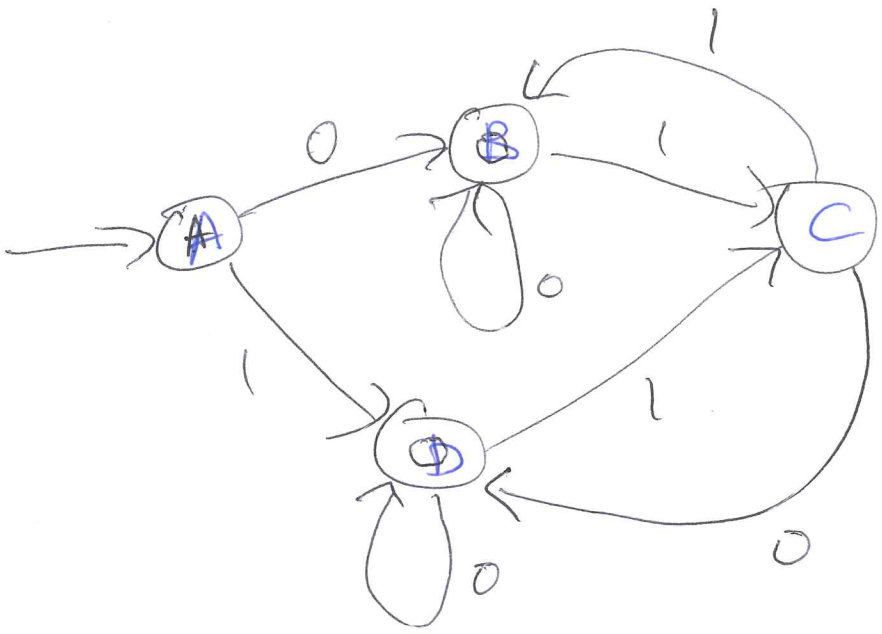
I can similarly check if there are any strings accepted by M_2 but not M_1.

This checks if M_1 & M_2 are the same.

#14 reg lang w/o λ, construct a right-lin gram. with prod.

$A \to aB$,

$A \to a$



$A \to 0$

$A \to 1$

$B \to 0$

$D \to 0$

$C \to 0$

$C \to 1$