CSCE 355
1/17/2024

Product construction on DFAs

Example: string search

NFAs (?)

Last time: If L is regular, then $\overline{L}$ is regular

$$\overline{L(A)} = L(\neg A)$$

Today: Prove that if L_1 & L_2 are regular langs then $L_1 \cap L_2$ is regular.

Def: Let ~~DFAs~~

$$A_1 := \langle Q_1, \Sigma, \delta_1, q_1, F_1 \rangle$$
$$A_2 := \langle Q_2, \Sigma, \delta_2, q_2, F_2 \rangle$$

be any DFAs with common input alphabet Σ.
The product of A_1 and A_2 is the DFA

$$A := \langle Q_1 \times Q_2, \Sigma, \delta, (q_1, q_2), F_1 \times F_2 \rangle$$

where for any $q \in Q_1$ and $r \in Q_2$ and $a \in \Sigma$,

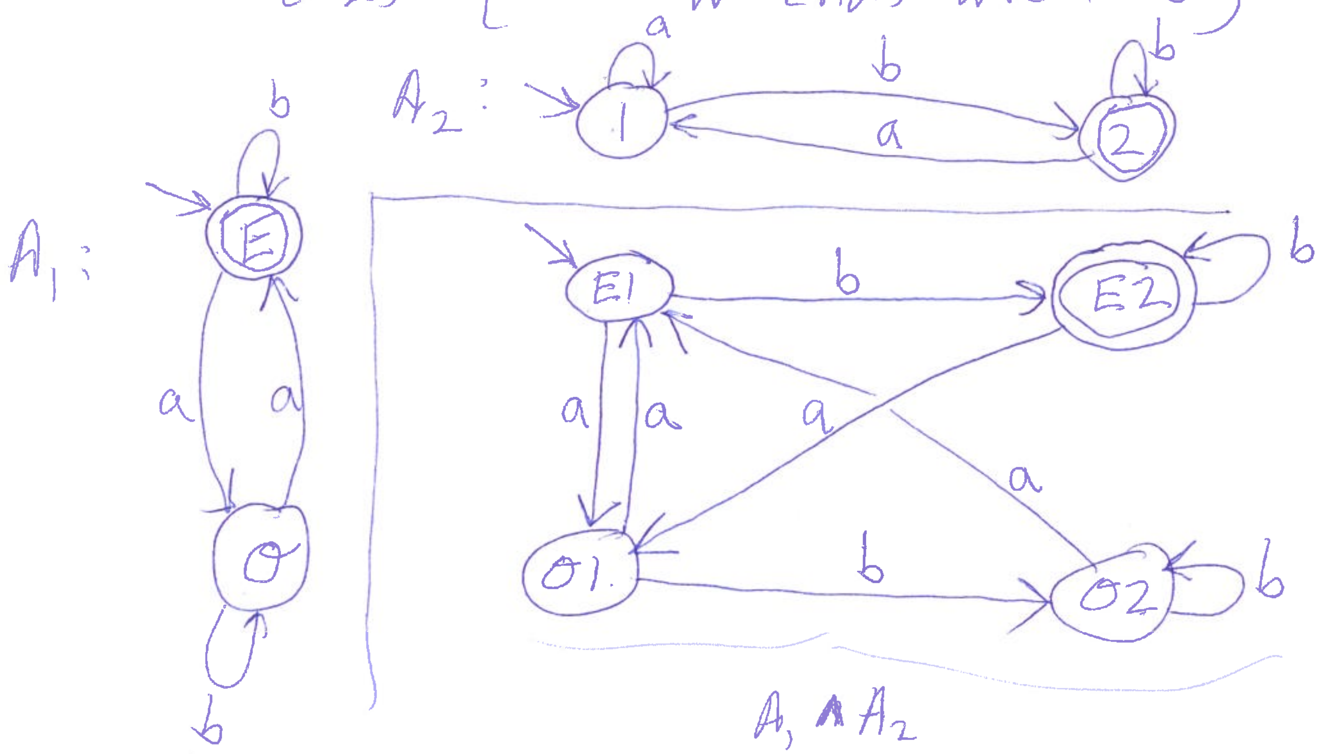
$$\delta((q,r), a) := (\delta_1(q,a), \delta_2(r,a)).$$

Notation: $A := A_1 \wedge A_2$.

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

$L(A_1) := \{w : w \text{ has an even number of } a\text{'s}\}$

$L(A_2) := \{w : w \text{ ends with } b\}$



<u>Thm</u>: A_1, A_2 as in the definition (DFAs with common input alphabet). Then $L(A_1 \wedge A_2) = L(A_1) \cap L(A_2)$

Proof ~~will~~ will be by induction on the length of an input string.

<u>Lemma</u>: Let $A_1, A_2, A := A_1 \wedge A_2$ be as in the def. ~~Let~~ For input string $w \in \Sigma^*$, let $A_1(w) \in Q_1$ be the end state of the computational trace of A_1 on input w. ~~Simila~~ Define $\underset{\in Q_2}{A_2(w)}$, $\underset{\in Q_1 \times Q_2}{A(w)}$ similarly

Then $A(w) = (A_1(w), A_2(w))$.

Proof: By induction on $|w|$:

Base case: $|w| = 0$. Then $w = \varepsilon$.

Then $A(w) = A(\varepsilon) = \underbrace{(q_1, q_2)}_{\text{start state of } A} = (A_1(\varepsilon), A_2(\varepsilon))$

$= (A_1(w), A_2(w))$. // base case

Inductive case: $|w| > 0$. There exist unique $x \in \Sigma^*$ and unique symbol $a \in \Sigma$ such that $w = xa$ and $|x| = |w| - 1 < |w|$

[x = principal prefix of w
a = last symbol of w]

Can assume (inductive hypothesis) that

$$A(x) = (A_1(x), A_2(x))$$

Then

$$A(w) \overset{w=xa}{=} A(xa) \overset{\text{"clear"}}{=} \delta(A(x), a)$$

$$\overset{\text{ind. hyp.}}{=} \delta((A_1(x), A_2(x)), a) \underset{\text{def. of } \delta}{=} (\delta_1(A_1(x), a), \delta_2(A_2(x), a))$$

"clear" fact applied to A_1 and A_2

$$\overset{\downarrow}{=} \big(A_1(xa), A_2(xa)\big) \underset{w=xa}{\overset{}{=}} (A_1(w), A_2(w))$$

$\therefore$ Lemma holds for w.

$\therefore$ Conclude by induction that the lemma holds for all $w \in \Sigma'^*$. $\boxed{//}$ lemma

<u>Proof of the theorem</u>: Let $w \in \Sigma'^*$ be arbitrary. W.T.S. that $w \in L(A) \iff w \in L(A_1) \cap L(A_2)$

[conclude that $L(A) = L(A_1) \cap L(A_2)$, where $A = A_1 \wedge A_2$]

$$w \in L(A) \underset{\text{def. of } L(A)}{\iff} A \text{ accepts } w$$

def. of acceptance in A

$$\iff A(w) \in F_1 \times F_2$$

($A(w)$ is accepting state of A)

by the lemma

$$\iff (A_1(w), A_2(w)) \in F_1 \times F_2$$

def of cartesian product

$$\iff A_1(w) \in F_1 \text{ and } A_2(w) \in F_2$$

def of acceptance in A_1 and in A_2

$$\iff A_1 \text{ accepts } w \text{ and } A_2 \text{ accepts } w$$

def of $L(A_1)$ & $L(A_2)$

$$\iff w \in L(A_1) \text{ and } w \in L(A_2)$$

def of $\cap$

$$\iff w \in L(A_1) \cap L(A_2)$$

<u>done</u> $\boxed{//}$

<u>Cor</u>: If L_1 & L_2 are regular langs, then $L_1 \cap L_2$ is regular, equiv: REG_Σ is closed under $\cap$.

<u>Proof</u>: By the prev. thm.

<u>Cor</u>: If L_1 & L_2 are regular, then $L_1 \cup L_2$ is regular.

<u>Proof</u>: By DeMorgan's laws,

$$L_1 \cup L_2 = \overline{\overline{L_1} \cap \overline{L_2}}.$$

Similarly, if L_1, L_2 are regular then so are

$$L_1 - L_2 := \{w : w \in L_1 \text{ \& } w \notin L_2\} = L_1 \cap \overline{L_2}$$

(<u>relative complement</u> of L_2 in L_1)

$$\begin{aligned} L_1 \triangle L_2 &:= (L_1 - L_2) \cup (L_2 - L_1) \\ &= \{w : w \in L_1 \text{ or } w \in L_2 \text{ but not both}\} \\ &= (L_1 \cup L_2) - (L_1 \cap L_2) \end{aligned}$$

(symmetric difference of L_1 & L_2)

<u>Note</u>: $L_1 = L_2$ iff $L_1 \triangle L_2 = \emptyset$.

Practical application of DFAs: string search

Given a fixed string x want to know if x appears as a substring of input string w.

Def: $x, w \in \Sigma^*$.

x is a prefix of w if $\exists y \in \Sigma^*$, $w = xy$

x is a suffix of w if $\exists y \in \Sigma^*$, $w = yx$

x is a substring of w if $\exists y, z \in \Sigma^*$,

$$w = yxz$$

(equiv. x is a prefix of some suffix of w)

Given a fixed "search" string x ~~and~~ we build a DFA S_x that on ~~is~~ input w accepts iff w has x as a substring.

Example: $\Sigma = \{a, b, c\}$, $x = abacaba$

S_x: states ($8 = |x| + 1$ of them) are labeled with the prefixes of x:

S_x: ε a ab aba $abac$ $abaca$ $abacab$ x

Idea: S_x is is state $y \in \Sigma^*$ just when y is the longest prefix of x which is a suffix of what has been read so far.