Closure Properties of Regular Languages

Introduction

Closure properties

- Closure properties let us compose languages
- Useful to specify / model

Outline

- Reverse
- Complement
- Intersection
- Difference

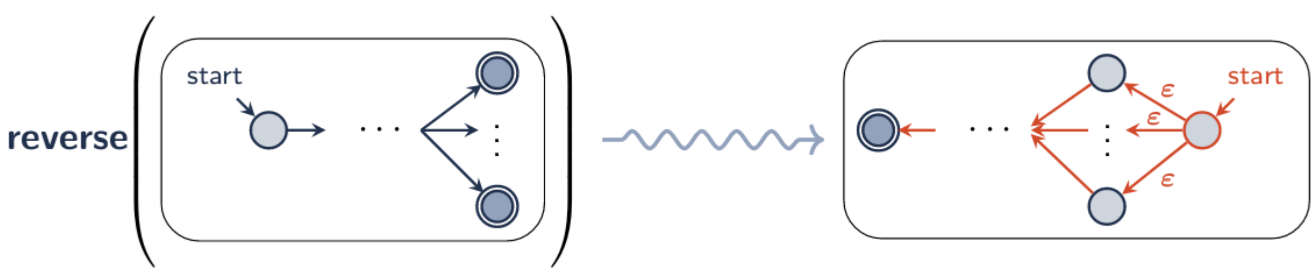
Reverse

Input: Finite Automaton A

Output: Finite Automaton R where $L(\mathrm{R}) = \{\sigma_0 \ldots \sigma_\mathrm{n} \mid \sigma_\mathrm{n} \ldots \sigma_0 \in L(\mathrm{A})\}$

Solution

1. Reverse edges of A
2. Start state of A becomes new accept state
3. New start state with ϵ transitions to old accept states



Reverse Algorithm

Function fa-reverse(A)

Input: $A = (Q, \Sigma, E, q_0, F)$

Output: $A' = (Q', \Sigma', E', q_0{}', F')$

$\Sigma' \leftarrow \Sigma$;

$q_0{}' \leftarrow \text{newstate}()$;

$Q' \leftarrow Q \cup \{q_0{}'\}$;

$F' \leftarrow \{q_0\}$;

$$E' \leftarrow \underbrace{\left(\bigcup_{q_i \xrightarrow{\sigma} q_j \in E} q_j \xrightarrow{\sigma} q_i \right)}_{\text{Reverse Edges}} \cup \underbrace{\left(\bigcup_{q \in F} q_0{}' \xrightarrow{\varepsilon} q \right)}_{\text{New start to old accept}} ;$$

Reverse on Regular Expressions

Input: Regular Expression A

Output: Reverse of A, denoted as A^R

Solution: Inductive Construction

Basis: For symbol α being a $\in \Sigma$, ϵ, or $\emptyset$:

$\alpha^R = \alpha$

Induction: For α^R in:

Concatenation: $(\beta\gamma)^R = \gamma^R\beta^R$

Union: $(\beta|\gamma)^R = \beta^R \mid \gamma^R$

Kleene-closure: $(\beta*)^R = (\beta^R)*$

Regex Reverse Algorithm

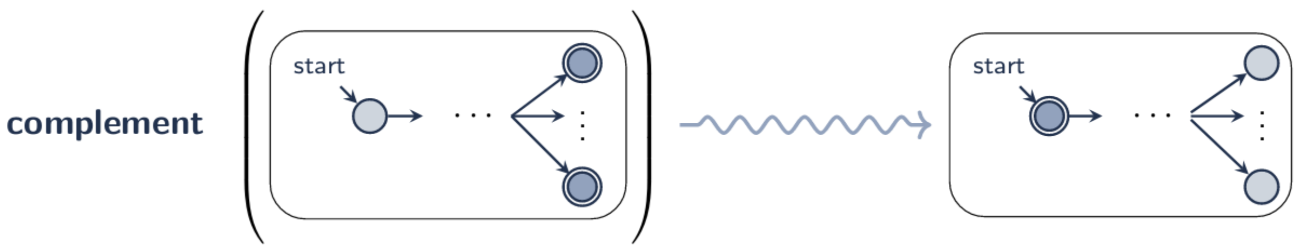
```
Function regex-reverse(e)
if e ∈ (Σ ∪ {ε, ∅}) then // Basis
    return e;
else // Induction
    switch car(e) do
        case UNION do
            return cons(UNION, map (regex-reverse, cdr (e)))
        case CONCATENATION do
            return cons(CONCATENATION, map (regex-reverse, reverse(cdr (e))))
        case KLEENE-CLOSURE do
            return list(KLEENE-CLOSURE, regex-reverse(second(e)))
```

Complement

Input: Regular Languages A

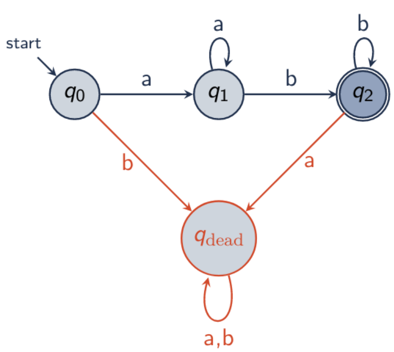
Output: $\bar{A} = \{\sigma \mid \sigma \notin \text{A}\}$

Solution: Flip accept / non-accept states



Dead States

Need to include dead states when taking the complement as they become accepts states in the conversion process.



Intersection

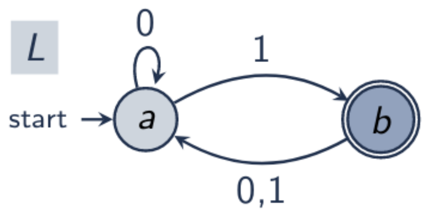
Input: Regular Languages A = $L(\mathrm{M_A})$ and B = $L(\mathrm{M_B})$

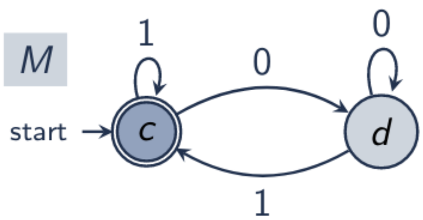
Output: C = $\{\sigma \mid (\sigma \in \mathrm{A}) \land (\sigma \in \mathrm{B})\}$

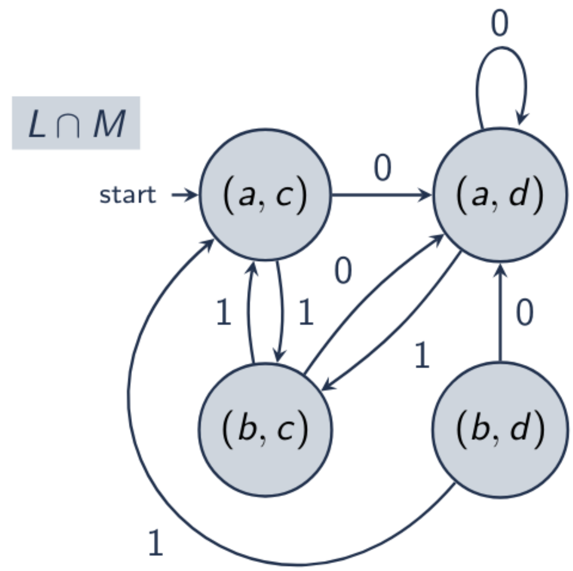
Solution: Use the product DFA to find C = $L(\mathrm{M_C})$

$\mathrm{M_C}$ accepts when both $\mathrm{M_A}$ and $\mathrm{M_B}$ accept

$\mathrm{F_C} = \{(\mathrm{q_i}, \mathrm{q_j}) \in \mathrm{Q_C} \mid \mathrm{q_i} \in \mathrm{F_A} \land \mathrm{q_j} \in \mathrm{F_B}\}$







Intersection Algorithm

Function dfa-intersect(M_1, M_2)

Input: $M_1 = (Q_1, \Sigma_1, E_1, q_{(0,1)}, F_1)$

Input: $M_2 = (Q_2, \Sigma_2, E_2, q_{(0,2)}, F_2)$

Output: $M' = (Q', \Sigma', E', q_0', F')$

$\Sigma' \leftarrow \Sigma_1 \cup \Sigma_2$;

$Q' \leftarrow Q_1 \times Q_2$; // Cartesian product of states

$q_0' \leftarrow \left(q_{(0,1)}, q_{(0,2)}\right)$;

$F' \leftarrow \{(q_i, q_j) \in Q' \mid q_i \in F_1 \land q_j \in F_2\}$; // accept in both M_1 and M_2

$E' \leftarrow$

$$\left\{ \underbrace{\left(q_{(i,1)}, q_{(i,2)}\right) \xrightarrow{\sigma} \left(q_{(j,1)}, q_{(j,2)}\right)}_{\text{Cartesian product edge}} \;\middle|\; \underbrace{\left(q_{(i,1)} \xrightarrow{\sigma} q_{(j,1)} \in E_1\right)}_{\sigma\text{-edge in } M_1} \land \underbrace{\left(q_{(i,2)} \xrightarrow{\sigma} q_{(j,2)} \in E_2\right)}_{\sigma\text{-edge in } M_2} \right\};$$

For E’ you are recursively visiting the product states.

Intersection Visit Product States Algorithm

Algorithm 1: dfa-product

Input: $M_1 = (Q_1, \Sigma, \delta_1, q_{(0,1)}, F_1)$
Input: $M_2 = (Q_2, \Sigma, \delta_2, q_{(0,2)}, F_2)$
Output: Q', E'

function *visit*(q_1, q_2) **is**
    forall $\sigma \in \Sigma$ **do**
        let
            $q_1' \leftarrow \delta_1(q_1, \sigma)$;
            $q_2' \leftarrow \delta_2(q_2, \sigma)$;
            $q' \leftarrow ({q_1}', {q_2}')$;
        in
            if ${q_1}' \wedge {q_2}' \wedge (q' \notin Q')$ **then**
                $Q' \leftarrow Q' \cup \{q'\}$;
                $E' \leftarrow E' \cup \left\{(q_1, q_2) \xrightarrow{\sigma} q'\right\}$;
                `visit`$({q_1}', {q_2}')$;

`visit`$(q_{(0,1)}, q_{(0,2)})$;

Difference

Input: Finite Automaton M_A and M_B

Output: Finite Automaton M_D where

$L(M_D) = L(M_A)\ /\ L(M_B) = \{\sigma \mid \sigma\ L(M_A) \wedge \sigma \notin L(M_B)\}$

Solution: Use product DFA

Final states where M_A accepts and M_B does not

$F_D = \{(q_a, q_b) \in Q_D \mid (q_a \in F_A) \wedge (q_b \notin F_B)\}$

Difference: is the intersection of the DFA except one of them is the complement