LANGAGES FORMELS

ENS Paris-Saclay
DER informatique
2ème semestre 2025-26

Organisation du cours → https://samvangool.net/langform.html

- **Deux parties** de 6 semaines chacune :

  1. Langages reguliers, automates, monoïdes — 22 janvier – 12 mars

     Sam van Gool (cours) et Guillaume Scerri (TD) → CCTD1

     → Partiel : 19 mars, 14h, salle 1Z61 (!)

  2. Langages algebriques, analyse syntaxique, langages d'arbres — 26 mars – 11 mai

     Stefan Schwoon (cours) et Luc Lapointe (TD) → CCTD2

     → Examen : semaine du 18 mai

     → Projet 2ème partie

- **Conditions de validation** :
  - Session 1 : CCTD1 (1), CCTD2 (1), projet (1), partiel (2), examen (2)
  - Session 2 : " , " , " , examen (4)

1. Words, languages, automata

The basic building blocks.

The rest of the slides are written in English, because this way, your lecturer will write less (fewer?!) grammar mistakes.
We will speak French in class. If any of this causes (informal) language issues, don't hesitate to tell me.
I hope you'll come to appreciate the mix as a feature, not a bug!

Let Σ be a finite set, which we call the alphabet.

A word is a finite sequence of symbols from Σ. The set of words is denoted Σ^*.

Example. - $\Sigma = \{0,1\}$. Words are binary sequences, e.g., 1001, 01001, 000.

! - $\Sigma = \{a, b, ab\}$. The expression abab does not define a word in this alphabet.

We would need to write (a,b,ab) or (ab,ab) or (ab,a,b).

$\Rightarrow$ We usually avoid taking such sets as alphabets, and assume unique parsability without commas.

We usually write ε for the empty word, i.e., the unique sequence of length 0. $\Sigma^+ := \Sigma^* \setminus \{\varepsilon\}$.

Given words $u, v \in \Sigma^*$, we can form their concatenation, $u \cdot v$. We often omit the $\cdot$.

We write $|u|$ for the length of u, defined inductively: $|\varepsilon| := 0$, $|ua| := |u| + 1$ for all $u \in \Sigma^*, a \in \Sigma$.

Fact. The structure $(\Sigma^*, \cdot, \varepsilon)$ is a monoid, which is free over Σ.

This means: for all $u, v, w \in \Sigma^*$, $(u \cdot v) \cdot w = u \cdot (v \cdot w)$, $u \cdot \varepsilon = u = \varepsilon \cdot u$, (monoid)

and any function $\Sigma \to M$, with M a monoid, has a unique homomorphic extension $\Sigma^* \to M$ (free).

Let $w \in \Sigma^*$.

A **prefix** of w is a word $u \in \Sigma^*$ such that
there exists $v \in \Sigma^*$ such that $w = uv$.

A **suffix** of w is a word $v \in \Sigma^*$ such that
there exists $u \in \Sigma^*$ such that $w = uv$.

A **factor** of w is a word $f \in \Sigma^*$ such that
there exist $u, v \in \Sigma^*$ such that $w = ufv$.

A **subword** of w is a word $s = (s_1, \dots, s_n) \in \Sigma^*$ $(n \geq 0)$ such that
there exist $u_0, u_1, \dots, u_n \in \Sigma^*$ such that $w = u_0 s_1 u_1 \dots s_n u_n$.

A **language** over the alphabet Σ is a subset of Σ^*.

Formal language theory is the study of the set $\mathcal{P}(\Sigma^*)$ of languages.

Fact. There are uncountably many languages (if $\Sigma \neq \emptyset$, which we will always assume).

(Exercise) → By this I mean: "If you don't know how to prove it, please try, or look it up online, or in a book, or ask a friend, or ask me, or..."

I do NOT mean: "This is easy and you should feel bad if you find it difficult."

(We just don't have time to discuss everything.)

G. Cantor (1845-1918)

⇒ We need methods of describing some of these languages, at least.

Automata and **regular expressions** are such methods.

They are fundamental to syntactic analysis, logic, verification, and more.

photo credit ↴
http://www.math.uni-hamburg.de/home/grothkopf/fotos/math-ges/

An automaton over Σ is a directed multigraph with Σ-labeled edges and two distinguished subsets.

Explicitly, an automaton is a quintuple $A = (Q, \Sigma, \delta, I, F)$, where

- Q is a set, whose elements are called states;
- Σ is an alphabet;
- $\delta \subseteq Q \times \Sigma \times Q$, its elements are called transitions or edges;
- $I \subseteq Q$ a set of initial states;
- $F \subseteq Q$ a set of final states.

We often denote an edge (q, a, r) as $q \xrightarrow{a} r$. Its source is q, its target is r, its label is a.

We usually assume (without saying so) that Q is finite.

Acronym: NFA ("AFN" en français)

An automaton $A = (Q, \Sigma, \delta, I, F)$ accepts a word $w \in \Sigma^*$ if there exists a path from an initial state to a final state that is labeled by w. Otherwise, A rejects w.

Explicitly, A accepts w provided that there exists a sequence $\pi \in Q^*$ with $|\pi| = |w| + 1$ such that:

- $\pi_0 \in I$,
- $\pi_{|w|} \in F$,
- for each $0 \leq i < |w|$, $\pi_i \xrightarrow{w_i} \pi_{i+1}$ in δ

$\Bigg\}$ π is a successful run on w

The language recognized by A is $\mathcal{L}(A) := \{ w \in \Sigma^* \mid A \text{ accepts } w \}$.

A language $L \subseteq \Sigma^*$ is recognizable if there exists an NFA that recognizes it, and we put

$$\mathrm{Rec}(\Sigma^*) := \{ L \subseteq \Sigma^* \mid L \text{ recognizable} \}.$$

Question. When does ε belong to $\mathcal{L}(A)$?

By this, I mean: "I think somebody will have a reasonable guess. Please answer me to avoid long awkward silences!"

↳ a.k.a.: "complete and deterministic" or "DFA"

An automaton $A = (Q, \Sigma, \delta, I, F)$ is deterministic if, for each $a \in \Sigma$, the relation $\delta_a := \delta \cap (Q \times \{a\} \times Q)$ is functional and total, and $\# I = 1$.

That is, for every $q \in Q$, $a \in \Sigma$, there exists a unique $r \in Q$ such that $q \xrightarrow{a} r$.

We sometimes write $q \cdot a$ for this unique state.

The transition relation of a DA can also be viewed as a function $\cdot : Q \times \Sigma \to Q$, or as a function $\rho : \Sigma \longrightarrow Q^Q$.

Fact. For any set Q, the triple $(Q^Q, \circ, id_Q)$ is a monoid.

Thus, by the free property of Σ^*, there exists a unique homomorphic extension $\hat{\rho} : \Sigma^* \longrightarrow Q^Q$ of ρ.

Explicitly, $\hat{\rho}$ can be defined by induction: $\hat{\rho}(\varepsilon) := id_Q$, and, for any $w \in \Sigma^*$, $a \in \Sigma$,

$\hat{\rho}(wa) = \lambda q . \rho(a)(\hat{\rho}(w)(q))$. ("the function sending $q \in Q$ to: $\rho(a)$ applied to $\hat{\rho}(w)(q)$").

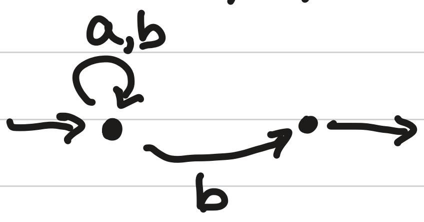
We also write $\tilde{\delta}(w) := q_0 \cdot w$, where q_0 is the initial state.

Examples.

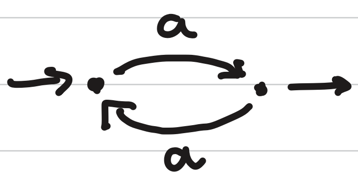
$\to \bullet$: initial

$\bullet \to$: final

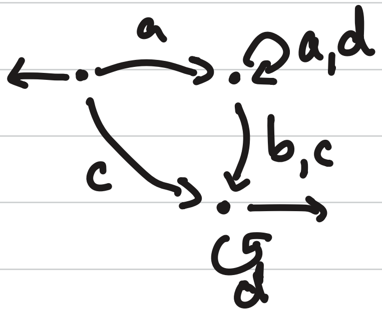
① $\Sigma = \{a, b\}$



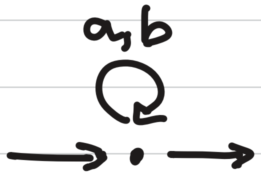
② $\Sigma = \{a\}$



③ $\Sigma = \{a, b, c, d\}$



④ $\Sigma = \{a, b\}$



Questions. - What is $\mathcal{L}(A)$?

- Are there other automata recognizing the same language? How many? Bigger? Smaller?

- Is A deterministic? If not, can we make it so, without changing $\mathcal{L}(A)$?

2. Determinization

Constructing an equivalent DFA out of an NFA using the power set.

Theorem. Any recognizable language can be recognized by a deterministic automaton.

Proof. Let $A = (Q, \Sigma, \delta, I, F)$ be an automaton recognizing L.

Define the automaton $\mathcal{P}(A) := (\mathcal{P}Q, \Sigma, \Delta, \{I\}, G)$, where:

- $\mathcal{P}Q$ is the set of subsets of Q (the power set of Q)
- $\Delta := \{(S, a, T) \in \mathcal{P}Q \times \Sigma \times \mathcal{P}Q \mid T = \{q \in Q \mid \exists s \in S, (s, a, q) \in \delta\}\}$
- $G := \{S \in \mathcal{P}Q \mid S \cap F \neq \emptyset\}$.

Note: $\mathcal{P}(A)$ is deterministic.

(Proof of Theorem, p.2)

We now claim that $L(A) = L(\mathbb{P}(A))$.

Proof of claim. We will show by induction that, for any $w \in \Sigma^*$,

$R(w) :=$

$\tilde{\Delta}(w) = \{ q \in Q \mid$ there exists a w-path in A such that $\pi_0 \in I$ and $\pi_{|w|} = q \}$

Base case. $w = \varepsilon$: both sets are equal to I.

Inductive case. $w = ua$ for $u \in \Sigma^*$, $a \in \Sigma$. By the induction hypothesis, $\tilde{\Delta}(u) = R(u)$.

Let $q \in Q$. $q \in \tilde{\Delta}(w) \iff$ there exists $s \in \tilde{\Delta}(u)$ such that $(s,a,q) \in \delta$ (def. of Δ)

$\iff$ there exists a u-path π s.t. $\pi_0 \in I$ and $\pi_{|u|} = s$ and $(s,a,q) \in \delta$ (def of R)

$\iff$ $q \in R(w)$ (def of w-path). $\square$ Now, $w \in L(\mathbb{P}(A)) \iff \tilde{\Delta}(w) \in G$

$\iff R(w) \cap F \neq \emptyset \iff w \in L(A)$.

$\square$ $\square$

The theorem gives a determinization construction.

Note that $\#\mathcal{P}(Q) = 2^{\#Q}$, so it has exponential cost, at worst.

Question. Is there a "better" determinization?

Example.
$$\begin{tikzcd} {} \arrow[r] & \bullet \arrow[loop above, "{a,b}"] \arrow[r, "a"] & \bullet \arrow[r, "{a,b}"] & \bullet \arrow[r, "{a,b}"] & \bullet \arrow[r, "{a,b}"] & \bullet \arrow[r, "{a,b}"] & \bullet \arrow[r] & {} \end{tikzcd}$$
recognizes $\{a,b\}^* \cdot \{a\} \cdot \{a,b\}^4$.

We get 2^6 states if we determinize.

More generally, $L_n := \{a,b\}^* \cdot \{a\} \cdot \{a,b\}^n$ can be recognized by an NFA with $n+2$ states.

However: Any deterministic automaton recognizing L_n requires at least 2^n states.

(We will see a proof of this later.)

The membership problem takes as input an automaton $A = (Q, \Sigma, \delta, I, F)$ and a word $w \in \Sigma^*$, and asks to determine whether or not $w \in \mathcal{L}(A)$.

Algorithm. We will keep track of a variable W which contains the reachable states after reading w.

1) $W \leftarrow I$
2) while $w \neq \varepsilon$:
3) $\quad$ new $\leftarrow \emptyset$
4) $\quad$ for all $q \in W$:
5) $\quad\quad$ new $\leftarrow$ new $\cup \{ r \in Q \mid (q, \mathrm{head}(w), r) \in \delta \}$
6) $\quad W \leftarrow$ new
7) $\quad w \leftarrow \mathrm{tail}(w)$
8) return $(W \cap F \neq \emptyset)$

Runtime: $O(|w| \cdot (\#Q)^2)$.

When A is known to be deterministic, we can achieve $O(|w|)$ (exercise).

The emptiness problem asks, given an automaton A, whether or not $\mathcal{L}(A) = \emptyset$.

The universality problem " " " " " " " " $\mathcal{L}(A) = \Sigma^*$.

For a DFA A, first compute Reach $:= \{ q \in Q \mid$ there exist $i \in I$ and a path $i \to q \}$.

(how? complexity?)

- $\mathcal{L}(A) = \emptyset \iff F \cap \text{Reach} = \emptyset$
- $\mathcal{L}(A) = \Sigma^* \iff \text{Reach} \subseteq F$.

For an NFA, emptiness can be done in the same way.

universality is PSPACE-complete.

3. Closure properties

Building recognizable languages via automata constructions

Theorem. Let $L \in \mathrm{Rec}(\Sigma^*)$. The complement $\Sigma^* \setminus L$ is recognizable, too.

Proof. Pick $(Q, \Sigma, \delta, I, F)$ a DFA recognizing L. (This exists thanks to determinization!)

The DFA $(Q, \Sigma, \delta, I, Q \setminus F)$ recognizes $\Sigma^* \setminus L$.

Indeed, for $w \in \Sigma^*$, $w \in \Sigma^* - L \iff \tilde{\delta}(w) \notin F$. $\square$

Theorem. Let $L_1, L_2 \in \mathrm{Rec}(\Sigma^*)$. The intersection $L_1 \cap L_2$ is recognizable, too.

Proof. Pick $A_i = (Q_i, \Sigma, \delta_i, I_i, F_i)$ an NFA recognizing L_i, for $i = 1, 2$.

Define the automaton $A_1 \times A_2 := (Q_1 \times Q_2, \Sigma, \delta, I_1 \times I_2, F_1 \times F_2)$

where $\delta := \{ ((q_1, q_2), a, (r_1, r_2)) \in (Q_1 \times Q_2) \times \Sigma \times (Q_1 \times Q_2) \mid$

$q_1 \xrightarrow{a} r_1$ in A_1 and $q_2 \xrightarrow{a} r_2$ in $A_2 \}$.

Then (claim!) $L(A_1 \times A_2) = L(A_1) \cap L(A_2)$.

Pf of claim (idea). An induction on $w \in \Sigma^*$ shows that a w-path in $A_1 \times A_2$ is precisely given by a pair of w-paths in A_1 and A_2. $\square$ $\square$

Example. For Σ a finite alphabet, let $NA(\Sigma) := \{ w \in \Sigma^* \mid \{ a \in \Sigma \mid \exists p, w_p = a \} \neq \Sigma \}$.

"not all"

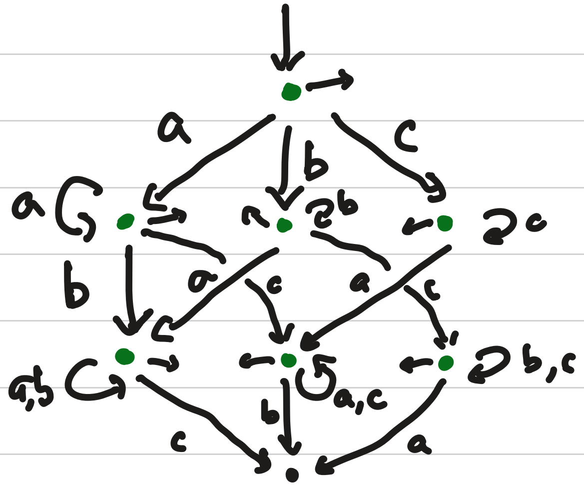
The automaton $(\Sigma, \Sigma, \delta, \Sigma, \Sigma)$ with

$$\delta = \{ (a, b, c) \in \Sigma \times \Sigma \times \Sigma \mid a \neq b,\ a = c \}$$

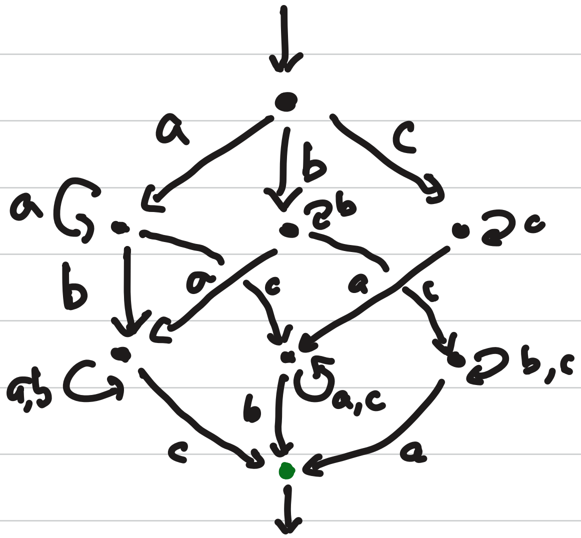
recognizes $NA(\Sigma)$.

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

determinize



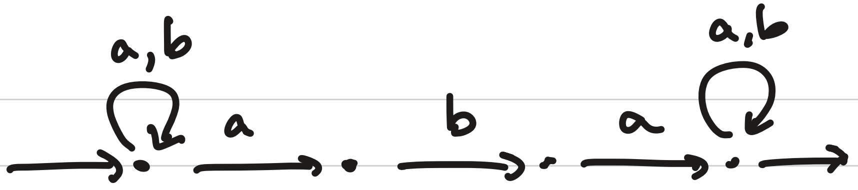
complement



Q Is there a smaller automaton than that one which recognizes $\Sigma^* - NA(\Sigma)$?

A smaller DFA? If not, how can we be sure?

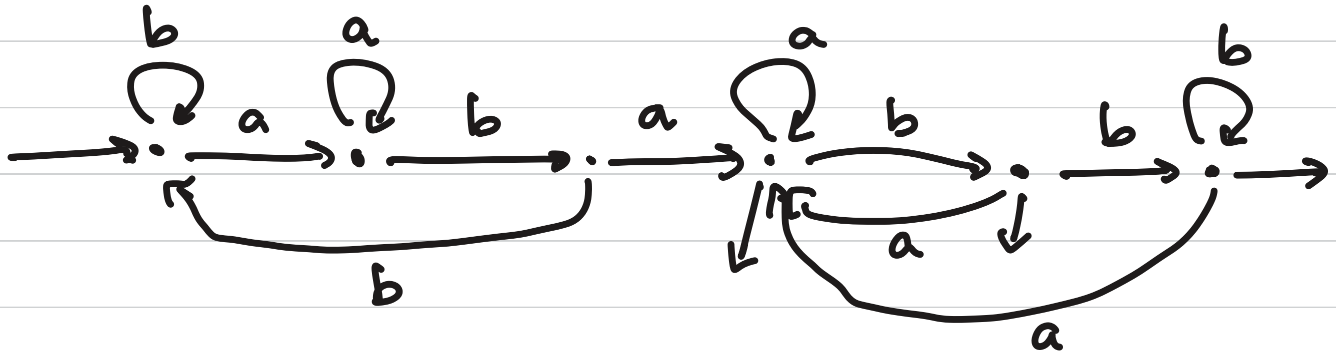
Example



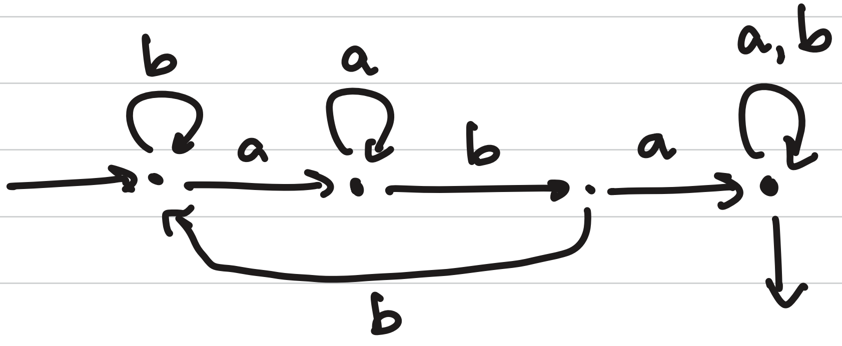
$L = \{ w \in \Sigma^* \mid aba \text{ is a factor of } w \}$

determinize $2^4 = 16$ states.

delete inaccessible states



collapse last 3 states



only 4 states!

For $L \in Rec(\Sigma^*)$, a DFA A such that $\mathcal{L}(A) = L$ is called **minimal** if, for every DFA B such that $\mathcal{L}(B) = L$, $\#Q_B \geq \#Q_A$.

We will prove later that every recognizable L has a unique minimal DFA.

Corollary. Let $L_1, L_2 \in Rec(\Sigma^*)$. Then $L_1 \cup L_2 \in Rec(\Sigma^*)$.

Proof 1. $L_1 \cup L_2 = \Sigma^* - ((\Sigma^* - L_1) \cap (\Sigma^* - L_2))$. $\square$

Proof 2. Let A_i be an automaton recognizing L_i $(i = 1, 2)$.

Define $A = A_1 \uplus A_2 = (Q_1 \uplus Q_2, \Sigma, \delta, I_1 \uplus I_2, F_1 \uplus F_2)$,

where $\delta := \{(q, a, r) \mid (q \in Q_1, r \in Q_1, \text{and } (q,a,r) \in \delta_1) \text{ or } (q \in Q_2, r \in Q_2, \text{and } (q,a,r) \in \delta_2\}$.

Since Q_1 and Q_2 are disconnected, if π is an accepting path in A starting in I_k, then it must end in F_k.

Thus, $L(A) = L(A_1) \cup L(A_2)$. $\square$

Question. Which proof do you prefer? Why?

Let $K, L \subseteq \Sigma^*$. The concatenation of K and L is

$$K \cdot L := \{ u \cdot v \mid u \in K, v \in L \}.$$

Let $w \in \Sigma^*$. The left quotient of L with respect to w is

$$w^{-1} L := \{ u \in \Sigma^* \mid wu \in L \},$$

and the right quotient of L w.r.t. w is

$$L w^{-1} := \{ u \in \Sigma^* \mid uw \in L \}.$$

The left and right residuals of L w.r.t. K are

$$K \backslash L := \bigcap_{w \in K} w^{-1} L \quad \text{and} \quad L / K := \bigcap_{w \in K} L w^{-1}$$

$$= \{ u \in \Sigma^* \mid \text{for all } w \in K,\ wu \in L \} \qquad = \{ u \in \Sigma^* \mid \text{for all } w \in K,\ uw \in L \}.$$

For $n \in \mathbb{N}$, we define inductively the power by $L^0 := \{\varepsilon\}$, and $L^{n+1} := L^n \cdot L$.

We define the Kleene star by $L^* := \bigcup_{n \geq 0} L^n$. Also $L^+ := \bigcup_{n \geq 1} L^n$ ($\neq L^* \setminus \{\varepsilon\}$ in general!)

Theorem. Let $L \in \mathrm{Rec}(\Sigma^*)$. For any $w \in \Sigma^*$, $w^{-1}L \in \mathrm{Rec}(\Sigma^*)$ and $Lw^{-1} \in \mathrm{Rec}(\Sigma^*)$.

Proof. Pick $A = (Q, \Sigma, \delta, I, F)$ an automaton for L. Define $A' = (Q, \Sigma, \delta, I', F)$, where

$$I' := \{ q \in Q \mid \text{there exist } q_0 \in I \text{ and a } w\text{-path from } q_0 \text{ to } q \}$$

Then, for any $u \in \Sigma^*$,

A' accepts u $\iff$ there exist $q \in I'$, $r \in F$, and a u-path from q to r

$\iff$ there exist $q_0 \in I$, $r \in F$, a w-path from q_0 to q, and a u-path from q to r

$\iff$ there exist $q_0 \in I$, $r \in F$, and a (wu)-path from q_0 to r

$\iff$ A accepts wu.

So $L(A') = w^{-1}L(A) = w^{-1}L$.

The statement about Lw^{-1} is proved similarly (exercise). $\square$

Corollary. Let $L \in \mathrm{Rec}(\Sigma^*)$. The set $\{ w^{-1}L : w \in \Sigma^* \}$ is finite.

Proof. Let A be an automaton such that $\mathcal{L}(A) = L$.
We showed in the previous proof that, for any $w \in \Sigma^*$, $w^{-1}L$ is recognized by a variant of A obtained by changing the inital states. Thus,
$\{ w^{-1}L : w \in \Sigma^* \} \subseteq \{ \mathcal{L}(A') \mid A' \text{ a initial-state-variant of } A \}$.
The second set has at most $2^{\# Q_A}$ elements. $\square$

Exercise. Let $L \in \mathrm{Rec}(\Sigma^*)$. For any $K \subseteq \Sigma^*$, $K \backslash L$ and L / K are recognizable.

Theorem. For any $K, L \in \mathrm{Rec}(\Sigma^*)$, $K \cdot L$ is recognizable.

For the proof, we will use a variant of automata.

Definition. An automaton with ε-transitions is a tuple $A = (Q, \Sigma, \delta, I, F)$, where Q, Σ, I, F are as in the definition of automaton, and $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$.

A accepts a word $w \in \Sigma$ if there exist $k_0, k_1, \ldots, k_{|w|} \in \mathbb{N}_{\geq 0}$ such that the NFA $A' := (Q, \Sigma \cup \{\varepsilon\}, \delta, I, F)$ accepts the word

$$\varepsilon^{k_0} w_1 \varepsilon^{k_1} \ldots \varepsilon^{k_{|w|-1}} w_{|w|} \varepsilon^{k_{|w|}}.$$

Equivalently: (exercise)

Let $w \in \Sigma^*$. A w-path in A is a finite word of edges $\pi \in \delta^*$ such that: (1) for every $0 \leq i < |\pi| - 1$, the target of π_i = the source of π_{i+1}

(2) the concatenation of the labels of π is equal to w

and acceptance is defined as for NFA's.

Fact. Any automaton with ε-transitions can be transformed into an automaton without ε-transitions that recognizes the same language.

Proof. See TD1. $\square$

Proof of Theorem. Let A and B be automata.

Construct the automaton with ε-transitions $C = (Q_C, \Sigma, \delta_C, I_A, F_B)$, where

$Q_C := Q_A \uplus Q_B$, and

$\delta_C := \delta_A \cup \delta_B \cup \{(q, \varepsilon, r) \mid q \in F_A, r \in I_B\}$.

Claim. $L(C) = L(A) \cdot L(B)$

Proof. Let $w \in \Sigma^*$. $w \in L(A) \cdot L(B) \overset{\text{def.}}{\iff}$ there exist $u \in L(A)$, $v \in L(B)$ such that $w = uv$

$\overset{\text{def.}}{\iff}$ there exist paths $\pi_1 : q_0 \xrightarrow{u} q_1$ in A, $\pi_2 : r_0 \xrightarrow{v} r_1$ in B with $q_0 \in I_A, q_1 \in F_A,$ $r_0 \in I_B$, $r_1 \in F_B$

why?

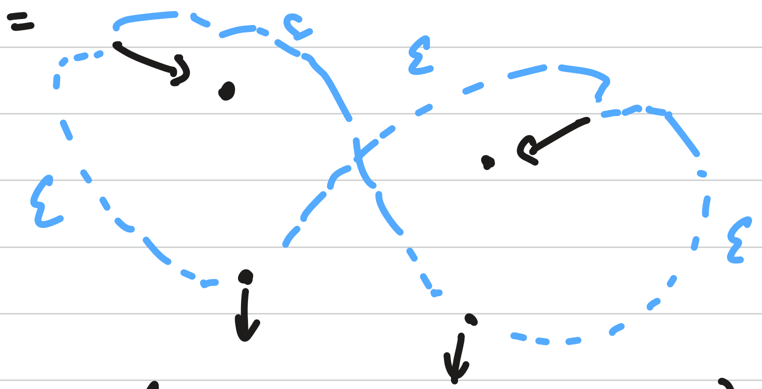
$\iff$ there exists w-path π in C starting in I_A, ending in F_B $\overset{\text{def.}}{\iff} w \in L(C)$. $\square$ $\square$

Theorem. For any $L \in Rec(\Sigma^*)$, $L^* \in Rec(\Sigma^*)$.

Idea

not a proof!

$A =$



Add ε-transition from any final to any initial state.

⚠ does not work ⚠

Example. $A = \bullet$ (one state, no edges). What does the construction above do?

An automaton is **normalized** if $\#I = \#F = 1$, the initial state is not a target of any edge, $I \cap F = \emptyset$, and the final state is not a source of any edge.

Lemma. For any automaton A, there exists a normalized automaton A' such that

$$L(A') = L(A) - \{\varepsilon\}.$$

Proof of Theorem. Let A be normalized recognizing $L - \{\varepsilon\}$. Define B by adding to A an ε-transition from the final state to the initial state, and making the initial state also final. Then $L(B) = L^*$ (exercise). $\square$

Proof of lemma. Given $A = (Q, \Sigma, \delta, I, F)$, define $A' = (Q', \Sigma, \delta', \{i_0\}, \{f_0\})$

where $Q' := Q \uplus \{i_0, f_0\}$

and $\delta' := \delta \cup \{(i_0, a, q) \mid \text{there exists } i \in I \text{ such that } i \xrightarrow{a} q \text{ in } A\}$

$\cup \{(q, a, f_0) \mid$ " " $f \in F$ " " $q \xrightarrow{a} f$ " " $\}$

$\cup \{(i_0, a, f_0) \mid \text{there exist } i \in I, f \in F \text{ such that } i \xrightarrow{a} f \text{ in } A\}$.

Then, for $w \in \Sigma^+$, we have $w \in L(A) \iff w \in L(A')$:

If $w \in L(A)$, let π be a successful run on w, from $i \in I$ to $f \in F$.

Since $w \neq \varepsilon$, $|\pi| = |w| + 1 \geq 2$. If $|w| = 1$, then, since π is successful, $i \xrightarrow{w} f$ in δ, so that (i_0, w, f) is in δ.

Suppose $|w| \geq 2$. Write $w = aw'b$, for $a, b \in \Sigma$ and $w' \in \Sigma^*$.

Let π' be defined by replacing the first node in π with i_0

and the last node in π with f_0. π' is a w-path in A'.

Conversely, if $w \in L(A')$, let π be a successful run in A'.
If $|w| = 1$ then there exist $i \in I, f \in F$ such that $i \xrightarrow{w} f$.
Otherwise, write $w = aub$ as before. Since π is successful, it begins with $i_0 \xrightarrow{a} q$ for some $q \in Q$, so we can pick $i \in Q$ such that $i \xrightarrow{w_0} q$.
Similarly, π ends with $r \xrightarrow{b} f_0$ for some $r \in Q$, so we can pick $f \in Q$ such that $r \xrightarrow{b} f$. Replacing i_0 with i and f_0 with f yields a successful run in A'. $\square$

4. Regular expressions

Describing recognizable languages syntactically.

Definition. A regular expression (r.e.) over alphabet Σ is an expression of one of the forms:

- ϕ,
- ε,
- for any $a \in \Sigma$: a,

or, for any regular expressions r_1, r_2:

- $r_1 \cdot r_2$,
- $r_1 + r_2$,
- $(r_1)^*$.

Application: $\boxed{\texttt{grep}}$

Same definition, written in Backus-Naur Form:

$$e ::= \phi \mid \varepsilon \mid a \mid e \cdot e \mid e + e \mid e^*$$

The language, $\mathcal{L}(e)$, of a r.e. e is defined inductively:

$\mathcal{L}(\phi) := \phi$, $\mathcal{L}(\varepsilon) := \{\varepsilon\}$, $\mathcal{L}(a) := \{a\}$, $\mathcal{L}(r_1 \cdot r_2) := \mathcal{L}(r_1) \cdot \mathcal{L}(r_2)$, $\mathcal{L}(r_1 + r_2) := \mathcal{L}(r_1) \cup \mathcal{L}(r_2)$, $\mathcal{L}(r_1^*) := \mathcal{L}(r_1)^*$.

A language L is regular if there exists a regular expression e such that $L = \mathcal{L}(e)$.

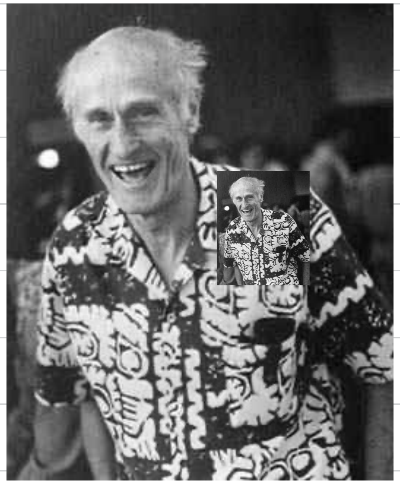
(Some people call this "rational". Terminology is difficult.)

Theorem (Kleene). A language $L \subseteq \Sigma^*$ is regular if, and only if, it is recognizable.

Proof. "$\Rightarrow$" By induction, for every regular expression e, we construct an automaton A_e such that $\mathcal{L}(e) = \mathcal{L}(A_e)$.

For the base cases, we have automata A_e:

e	$\emptyset$	ε	a
A_e	$\bullet$	$\to \bullet \to$	$\to \bullet \xrightarrow{a} \bullet \to$



S.C. Kleene (1909-1994)

In each case, $\mathcal{L}(A_e) = \mathcal{L}(e)$.

For the inductive cases, we apply the closure properties that we proved before.

$e = r_1 \cdot r_2$: Given A_{r_1} and A_{r_2}, construct B such that $\mathcal{L}(B) = \mathcal{L}(A_{r_1}) \cdot \mathcal{L}(A_{r_2})$.

Define $A_e := B$. Then $\mathcal{L}(A_e) = \mathcal{L}(A_{r_1}) \cdot \mathcal{L}(A_{r_2}) \overset{\text{IH}}{=} \mathcal{L}(r_1) \cdot \mathcal{L}(r_2) = \mathcal{L}(e)$

The cases $e = r_1 + r_2$ and $e = r_1^*$ are similar, using that recognizable languages are closed under union and star.

https://mathshistory.st-andrews.ac.uk/Biographies/Kleene/pictdisplay/

(Q,Σ,δ,I,F) → always, unless mentioned otherwise

Proof (continued). "$\Leftarrow$" Let A be an automaton. We construct r a regex such that $\mathcal{L}(r)=\mathcal{L}(A)$.

Let $n := \# Q$, and choose an (arbitrary) bijection $\{1,\dots,n\} \xrightarrow{\varphi} Q$. We write $q_i := \varphi(i)$, $1 \le i \le n$.

For each $0 \le k \le n$, $p,q \in Q$, define:

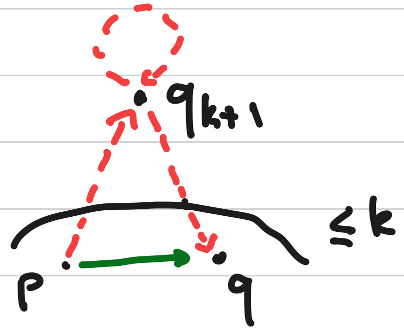
$$R^{(k)}_{p,q} := \{ w \in \Sigma^{+} \mid \text{there exists a } w\text{-path } \pi \text{ from } p \text{ to } q \text{ such that all states in } \pi \text{, except possibly the first and last, belong to the set } \{q_1,\dots,q_k\} \}.$$

Let $p,q \in Q$. We construct regexes $r^{(k)}_{p,q}$ such that $\mathcal{L}(r^{(k)}_{p,q}) = R^{(k)}_{p,q}$, for each $0 \le k \le n$.

Induction on k: $(0 \le k \le n)$

$k=0$. The only possible paths have length 1.

Let $r^{(k)}_{p,q} :=$ sum of $a \in \Sigma$ such that $(p,a,q) \in \delta$.



$k \Rightarrow k+1$. Note that $R^{(k+1)}_{p,q} = R^{(k)}_{p,q} + R^{(k)}_{p,q_{k+1}} \cdot \left(R^{(k)}_{q_{k+1},q_{k+1}} \right)^{*} \cdot R^{(k)}_{q_{k+1},q}$

So define $r^{(k+1)}_{p,q} := r^{(k)}_{p,q} + r^{(k)}_{p,q_{k+1}} \cdot \left(r^{(k)}_{q_{k+1},q_{k+1}}\right)^{*} \cdot r^{(k)}_{q_{k+1},q}$.

Finally, $r_{p,q} := \begin{cases} r^{(n)}_{p,q} & \text{if } p \ne q \\ r^{(n)}_{p,q} + \varepsilon & \text{if } p = q \end{cases}$, and $r :=$ sum of $r_{p,q}$ such that $p \in I$ and $q \in F$. $\square$

(+ Thompson, sometimes)

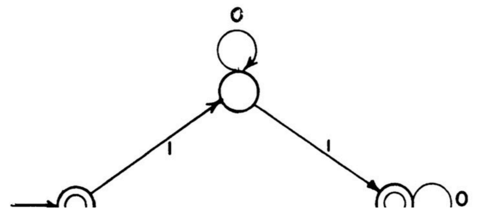
The algorithm used for "$\Leftarrow$" in the proof is called McNaughton-Yamada.

1960 IRE TRANSACTIONS ON ELECTRONIC COMPUTERS 39

Regular Expressions and State Graphs for Automata*

R. McNAUGHTON† AND H. YAMADA†

Summary—Algorithms are presented for 1) converting a state graph describing the behavior of an automaton to a regular expression describing the behavior of the same automaton (section 2), and 2) for converting a regular expression into a state graph (sections 3 and 4). These algorithms are justified by theorems, and examples are given. The first section contains a brief introduction to state graphs and the regular-expression language.



Other algorithms, sometimes with better outcomes, exist, notably, Brzozowski-McCluskey, which you will see in the TD, and the state elimination method, which relies on Arden's Lemma:

Lemma. Let $K, L \subseteq \Sigma^*$. The equation $X = K \cdot X + L$ has a smallest solution in $\mathcal{P}(\Sigma^*)$, namely, K^*L. If $\varepsilon \notin K$, the solution is unique.

Proof. See TD. $\square$

Proof of "$\Leftarrow$" in Kleene's theorem, using state elimination method.

Let A be an automaton. For every $q \in Q$, define $L_q := \{ w \in \Sigma^* \mid \text{there exist } f \in F \text{ and } q \xrightarrow{w} f \}$.

Note that $L_q = \bigcup_{r \in Q} \Sigma_{q,r} \cdot L_r \cup 1_{q,F}$, where

$\Sigma_{q,r} := \{ a \in \Sigma \mid (q,a,r) \in \delta \}$ and $1_{q,F} := \begin{cases} \varepsilon & \text{if } q \in F \\ \emptyset & \text{if } q \notin F \end{cases}$.

$a \in \Sigma_{q,r}$, $w' \in L_r$.

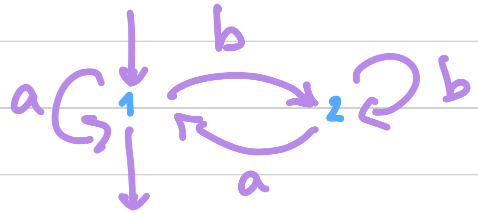
To obtain a regular expression, we successively solve the system of equations in $\#Q$ unknowns:

$$\begin{cases} R_1 = \sum_{i=1}^{n} \Sigma_{q_1,q_i} \cdot R_i + 1_{q_1,F} \\ \vdots \\ R_n = \sum_{i=1}^{n} \Sigma_{q_n,q_i} \cdot R_i + 1_{q_n,F} \end{cases},$$

using Arden's Lemma, in the same way as Gaussian elimination for systems of linear equations.

For instance, starting from the last equation, it has the form $X = KX + L$, where $X = R_n$, $K = \Sigma_{q_n,q_n}$, $L = \sum_{i=1}^{n-1} \Sigma_{q_n,q_i} \cdot R_i + 1_{q_n,F}$. So $R_n = K^* L$.

Example.



McNaughton – Yamada : $r^{(k+1)}_{p,q} := r^{(k)}_{p,q} + r^{(k)}_{p,q_{k+1}} \cdot \left(r^{(k)}_{q_{k+1},q_{k+1}}\right)^* \cdot r^{(k)}_{q_{k+1},q}$

$r^{(0)}_{1,2}$

$$r^{(0)} = \begin{pmatrix} a & b \\ a & b \end{pmatrix} \rightsquigarrow r^{(1)} = \begin{pmatrix} a + a \cdot a^* \cdot a & a + b \cdot a^* \cdot b \\ a + a \cdot a^* \cdot a & a + b \cdot a^* \cdot b \end{pmatrix} = \begin{pmatrix} a^+ & a^* b \\ a^+ & a^* b \end{pmatrix}$$

$$\rightsquigarrow r^{(2)} = \cdots = \begin{pmatrix} (a^* b)^* a^+ & (a^* b)^+ \\ (a^* b)^* a^+ & (a^* b)^+ \end{pmatrix}. \qquad r = r^{(2)}_{1,1} + \varepsilon = (a^* b)^* a^+ + \varepsilon.$$

$a + b \cdot b^* \cdot a$
$= a + b^+ a$
$= (\varepsilon + b^+) a$

Elimination :

$$\begin{cases} r_1 = a \cdot r_1 + b \cdot r_2 + \varepsilon \\ r_2 = a \cdot r_1 + b \cdot r_2 \end{cases} \rightsquigarrow \begin{cases} r_1 = a \cdot r_1 + b \cdot b^* \cdot a \cdot r_1 + \varepsilon = b^* a \cdot r_1 + \varepsilon \\ r_2 = b^* \cdot a \cdot r_1 \end{cases}$$

$$\rightsquigarrow \begin{cases} r_1 = (b^* a)^* \\ r_2 = b^* \cdot a \cdot (b^* a)^* = (b^* a)^+ \end{cases}$$

Another way to write this : $(a+b)^* a + \varepsilon$.

Example 2.6 in: J. Sakarovitch, Elements of Automata Theory, Cambridge University Press (2009), translated from French by R. Thomas

A **Kleene algebra** is a tuple $(K, 0, 1, +, \cdot, (\ )^*)$ where:

- $(K, 0, 1, +, \cdot)$ is a unital semiring, and

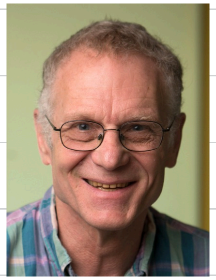
  ($+$ and $\cdot$ are associative, $+$ is commutative, $\cdot$ distributes over $+$, 0 is neutral for $+$ and 1 is neutral for $\cdot$.)

for all $a, b \in K$:

John H. Conway (1937–2020)

- $1 + a \cdot a^* = a^* = 1 + a^* a$
- If $b + a \cdot c \leq c$ then $a^* b \leq c$, and if $b + ca \leq c$ then $b a^* \leq c$

  where "$x \leq y$" means: "$x + y = y$".



Example. $(\mathrm{Rec}(\Sigma^*), \emptyset, \{\varepsilon\}, \cup, \cdot, (\ )^*)$ is a Kleene algebra.

Theorem. (Conway, Kozen) Let r, s be regular expressions. Then $\mathcal{L}(r) = \mathcal{L}(s)$ if, and only if, for every Kleene algebra K and $(k_a)_{a \in \Sigma} \in K^{\Sigma}$, $r[a \mapsto k_a] = s[a \mapsto k_a]$ in K.

Dexter Kozen

"Kleene algebras are a sound and complete axiomatization of regular languages."

Proof. Omitted. See: D. Kozen, "A completeness theorem for Kleene algebras ..." (1994).

https://en.wikipedia.org/wiki/John_Horton_Conway

https://www.cs.cornell.edu/~kozen/