TD 4: Regular expressions

Exercice 1: For each of the following languages, find an accepting regular expression:

1. the email addresses following the pattern <*firstname*>`.`<*lastname*>`@polytechnique.fr`
2. the Python language identifiers: «it starts with a letter (`A` ... `Z`) or (`a` ... `z`) or an underscore (_) followed by zero or more letters, underscores and digits (`0` ... `9`).»
3. the decimal representation of odd numbers (i.e. $2\mathbb{N} + 1$)
4. the binary words with even / odd occurences of 1.

Solutions:

1. $(a + \ldots + z)(a + \ldots + z)^*.(a + \ldots + z)(a + \ldots + z)^*@polytechnique.fr$
2. $(a + \ldots + z + A + \ldots + Z + \_)(a + \ldots + z + A + \ldots + Z + 0 + \ldots + 9 + \_)^*$
3. $(0 + \ldots + 9)^*(1 + 3 + 5 + 7 + 9)$
4. even: $(0^*10^*10^*)^*$
   odd: $(0^*10^*10^*)^*10^*$

Exercice 2: Let $\Sigma = \{0, 1\}$ and let L be the language of the regular expression $(1(01^*0)^*1 + 0)^*$

1. Find all the words of L of length at most 5
2. Give a «natural» description of the language L.
   (Hint: convert some words of L to their decimal representation, then make a guess)
3. Draw a finite automaton whose language is L? (Hint: starts from the inner most regular expression)

Solutions:

1.

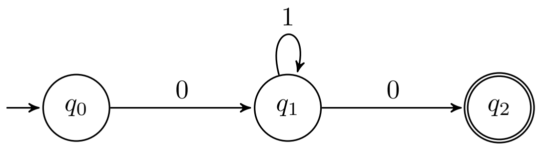
   ε 0 00 11 000 011 110 0000 0011 0110 1100 1111

   1001 00000 00011 00110 01100 11000 10101 01001 10010

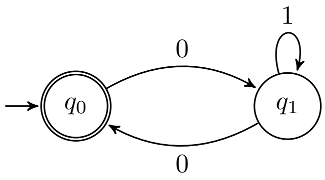
   01111 11011 11110

2. L is the set of all the multiples of 3 written in binary.
3. We proceed step by step, starting from the innermost regular expression.
   Warning: we are not strictly applying the algorithm seen in class (using ε-transitions, and then eliminating them). Instead, we simply "guess" step by step what the automaton for each sub-expression is, re-using the automaton constructed in the previous step.

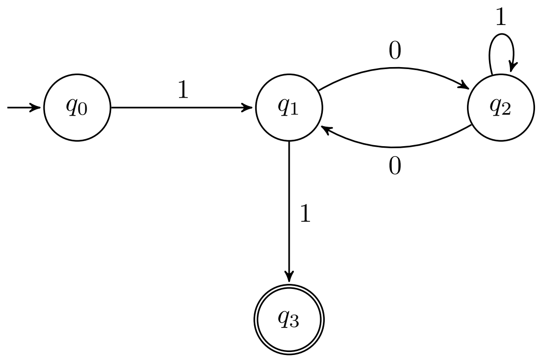
- 01^*0 is recognized by the following automaton:



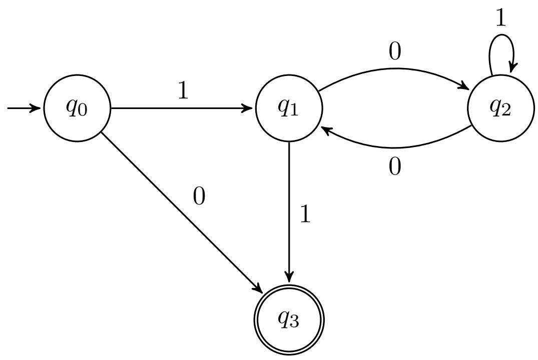
- $(01^*0)^*$ is recognized by the following automaton:



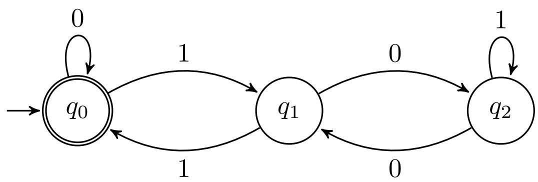
- $1(01^*0)^*1$ is recognized by the following automaton:



- $1(01^*0)^*1 + 0$ is recognized by the following automaton:



- $(1(01^*0)^*1 + 0)^*$ is recognized by the following automaton:



Warning: for the 2nd and 5th automata, to construct the star of the previous expression, we merged together the initial state and the final state. **This method does not work in general!** It works here because we have a unique initial state with no incoming transition, and a unique final state with no outgoing transition.

Exercice 3: A regular expression is said to be *star free* when the star operator does not appear in it.

1. Prove that the language matched by a star free regular expression is finite.
   (Hint: by structural induction on star free regular expressions.)

2. Find a star free regular expression e such that e^* is weakly equivalent to $(ba + a^*ab)^*a^*$.

Solutions:

1. Star-free regular expressions are produced by the following grammar: $E \to \varnothing \mid a \mid (E{+}E) \mid (EE)$, with $a \in \Sigma$. We prove the following property by structural induction on the set of star-free regular expressions: "for every star-free regular expression E, the language $[\![E]\!]$ is finite". There are 4 cases.

   - Case $E = \varnothing$: by definition, $[\![\varnothing]\!] = \emptyset$, which is a finite set.
   - Case $E = a$: by definition, $[\![a]\!] = \{a\}$, which is finite.
   - Case $E = (E_1 + E_2)$: by definition, $[\![(E_1 + E_2)]\!] = [\![E_1]\!] \cup [\![E_2]\!]$.
     By induction hypothesis on E_1, we know that $[\![E_1]\!]$ is finite.
     By induction hypothesis on E_2, we know that $[\![E_2]\!]$ is finite.
     Therefore, $[\![(E_1 + E_2)]\!]$ is finite because it is the union of two finite sets.
   - Case $E = (E_1E_2)$: by definition, $[\![(E_1E_2)]\!] = [\![E_1]\!] \cdot [\![E_2]\!]$.
     By induction hypothesis on E_1, we know that $[\![E_1]\!]$ is finite.
     By induction hypothesis on E_2, we know that $[\![E_2]\!]$ is finite.
     The concatenation of two finite languages is finite: indeed, by definition $L_1 \cdot L_2 = \{w_1w_2 \mid w_1 \in L_1 \text{ and } w_2 \in L_2\}$. If L_1 has n elements and L_2 has m elements, there can be at most $n \times m$ elements in $L_1 \cdot L_2$, so it is finite.
     Therefore, $[\![(E_1E_2)]\!]$ is finite because it is the concatenation of two finite sets.

2. We claim that $(ba + a^*ab)^*a^* \sim (a + ba + ab)^*$. Let us try to prove this claim (this is not part of the exercise!):

   - $[\![(ba + a^*ab)^*a^*]\!] \subseteq [\![(a + ba + ab)^*]\!]$:
     Let $w \in [\![(ba + a^*ab)^*a^*]\!]$. So, w can be decomposed as $w = w_1 \dots w_n a^k$, where each w_i is either $w_i = ba$ or of the form $w_i = a^{k_i}ab$. We want to show that $w \in [\![(a + ba + ab)^*]\!]$, i.e., that w can be decomposed as $w = w'_1 \dots w'_N$, where each w'_i is either a or ba or ab.

     For each component w_i of w, if $w_i = ba$, then it is already of the desired form. Otherwise, we have $w_i = a^{k_i}ab$ for some $k_i \in \mathbb{N}$, which we can decompose into k_i words of the form 'a' and one word of the form 'ab'. Finally, the suffix a^k of w can also be decomposed as desired.

   - $[\![(a + ba + ab)^*]\!] \subseteq [\![(ba + a^*ab)^*a^*]\!]$:
     This direction is a bit more tricky. Assume $w \in [\![(a + ba + ab)^*]\!]$, so $w = w_1 \dots w_n$ where each w_i is either a or ba or ab. We want to find a decomposition of w matching the pattern $(ba + a^*ab)^*a^*$.

     This is not easy to do: for example, suppose we are given the decomposition $w = a \cdot ba$. To show that $w \in [\![(ba + a^*ab)^*a^*]\!]$, we would need to split the 'ba' factor, in order to get $w = ab \cdot a$. Doing a general proof might be doable, but quite tedious!

Remark: fortunately, there is a general procedure to decide whether two regular expressions e and f recognize the same language:

- compute finite-state automata $\mathcal{A}_e$ and $\mathcal{A}_f$ recognizing the language of e and f, respectively
- compute the automata $\bar{\mathcal{A}}_e$ and $\bar{\mathcal{A}}_f$ recognizing the complement of these languages
- compute the automaton $\mathcal{A}_e \cap \bar{\mathcal{A}}_f$ and check that its language is empty (this implies $[\![e]\!] \subseteq [\![f]\!]$)
- compute the automaton $\bar{\mathcal{A}}_e \cap \mathcal{A}_f$ and check that its language is empty (this implies $[\![f]\!] \subseteq [\![e]\!]$)

Note that all these steps can be done by a computer: you can find plenty of software doing that, e.g.,
http://perso.ens-lyon.fr/damien.pous/symbolickat/
http://languageinclusion.org/doku.php?id=nfasimsub

Exercice 4: Let L be a language over an alphabet with a single element (i.e. $\Sigma = \{a\}$ and $L \subseteq \Sigma^*$).

1. Find a finite subset $F \subseteq L$ such that $L \subseteq F^*$. (Hint: F contains at most n elements, where w is the shortest nonempty word of L (if L is not empty) and n denotes the length of w.)
2. Deduce that L^* is regular.

Solutions: *Remark:* Here the language L is not assumed to be regular. To understand the exercise, it might be useful to keep an example in mind for L, to understand how the proof works. For instance, with $L = \{a^p \mid p \text{ is prime}\}$, the set that we want in the end is $F = \{a^2, a^3\}$.

1. If L is empty or $L = \{\varepsilon\}$, the set $F = \emptyset$ works.
   So, assume L is non-empty, and let w_0 be the shortest nonempty word of L. Since the alphabet contains only one letter, w_0 is of the form $w_0 = a^n$, where $n \geq 1$ is the size of w_0.
   Clearly, a^n has to be in F, otherwise we can never have $L \subseteq F^*$ since a^n cannot be obtained as a concatenation of longer words. First, let us check whether the set $F_0 = \{a^n\}$ works.

   - $F_0 \subseteq L$ by construction
   - but we might not have $L \subseteq F_0^*$

   If we are lucky and $L \subseteq F_0^*$, we are done.
   Otherwise, that means the set $L \setminus F_0^*$ is non-empty. Let w_1 be the shortest word of $L \setminus F_0^*$, and let $F_1 = F_0 \cup \{w_1\} = \{w_0, w_1\}$. As before, $F_1 \subseteq L$ by construction. Then, either $L \subseteq F_1^*$ and we are done, or the set $L \setminus F_1^*$ is non-empty and has a shortest element w_2.
   We can iterate this reasoning to construct a sequence of sets $(F_k)_{k \in \mathbb{N}}$. Formally, this is an inductive definition: when F_k has been defined, either $L \subseteq F_k^*$ which concludes the proof, or $L \not\subseteq F_k^*$, in which case we choose $F_{k+1} = F_k \cup \{w_{k+1}\}$ where w_{k+1} is the shortest word of $L \setminus F_k^*$.

   Now, how do we know that this procedure will eventually terminate?
   First, notice that $F_0^* = \{a^{qn} \mid q \in \mathbb{N}\}$. So, in F_0^*, we have all the words whose length is divisible by n. In particular, since w_1 is not in F_0^*, we know that $|w_1| \not\equiv 0 \mod n$. We write $k_1 = |w_1| \bmod n \in \{1, \dots, n-1\}$.
   Next, F_1^* contains all the words of L whose size is congruent to 0 or k_1 modulo n. Indeed, if $a^m \in L$ with $m \equiv k_1 \mod n$, then it has to be longer than w_1, and so we can decompose it as $a^m = w_1 \cdot (w_0)^q$ for some $q \in \mathbb{N}$. Therefore, the word w_2, which is not in F_1^*, must be such that $|w_2| \not\equiv 0 \mod n$ and $|w_2| \not\equiv k_1 \mod n$.
   If we iterate this reasoning, since there are only n distinct possible values for the remainder modulo n, we know that the set F_n^* will contain all the words of L. Thus, the set F_n works.

2. In question 1, we found a <u>finite</u> set F such that $F \subseteq L$ and $L \subseteq F^*$.

   Claim: if L_1 and L_2 are languages such that $L_1 \subseteq L_2$, then $L_1^* \subseteq L_2^*$.
   Proof: Let $w \in L_1^*$, then $w = w_1 \dots w_n$ where $w_i \in L_1$. Then we also have $w_i \in L_2$, so $w \in L_2^*$.

   Using the claim and the results of question 1, we get $F^* \subseteq L^*$ and $L^* \subseteq F^{**}$. But we also know that $F^{**} = F^*$ (see Exercise 5.2.(a)), so we have $L^* = F^*$. Since F is finite, it can be recognized by an automaton, and since regular languages are closed under Kleene star, F^* is regular.

Exercice 5: Let e, f, and g be regular expressions.

1. Prove the following equivalences (give a simple argument for each):

   (a) $e\varnothing \sim \varnothing \sim \varnothing e$ and $e\varnothing^* \sim e \sim \varnothing^* e$ and $e(fg) \sim (ef)g$

   (b) $e+\varnothing \sim e \sim \varnothing+e$ and $e+f \sim f+e$ and $e+e \sim e$ and $e+(f+g) \sim (e+f)+g$

   (c) $e(f+g) \sim ef+eg$ and $(e+f)g \sim eg+fg$

2. Prove the following equivalences (each requires a little more sophisticated argument):

   (a) $e^{**} \sim e^*$

   (b) $ee^* \sim e^*e$

   (c) $\varnothing^* + ee^* \sim e^*$

   (d) $(e+f)^* \sim (e^*f^*)^*$

Solutions: Remember that $e \sim f$ means (by definition) that $[\![e]\!] = [\![f]\!]$.

1.(a) Applying the definitions, the three equivalences become:

$$[\![e]\!] \cdot \emptyset = \emptyset = \emptyset \cdot [\![e]\!] \quad \text{and} \quad [\![e]\!] \cdot \emptyset^* = [\![e]\!] = \emptyset^* \cdot [\![e]\!] \quad \text{and} \quad [\![e]\!] \cdot ([\![f]\!] \cdot [\![g]\!]) = ([\![e]\!] \cdot [\![f]\!]) \cdot [\![g]\!]$$

The first two are trivially true (check the definition of the Kleene star if you are not sure why $\emptyset^* = \{\varepsilon\}$). The third one was done in Tutorial 1, Exercise 2, question 2.

1.(b) these equivalences boil down to the following facts about unions of sets:

$$A \cup \emptyset = A = \emptyset \cup A \quad \text{and} \quad A \cup B = B \cup A \quad \text{and} \quad A \cup A = A \quad \text{and} \quad A \cup (B \cup C) = (A \cup B) \cup C$$

1.(c) Done in Tutorial 1, Exercise 2, question 2.

2.(a) We want to prove: $([\![e]\!]^*)^* = [\![e]\!]^*$. For convenience, let us denote $L = [\![e]\!]$. We prove it by double inclusion.

- $L^* \subseteq L^{**}$: By definition $L^{**} = \bigcup_{n \in \mathbb{N}} (L^*)^n$, so for $n = 1$, we have $L^* \subseteq L^{**}$.
- $L^{**} \subseteq L^*$: Let $w \in L^{**}$. Then w can be decomposed in $w = w_1 w_2 \dots w_n$, with each $w_i \in L^*$. We can then decompose each w_i as $w_i = w_{i,1} \dots w_{i,k_i}$, with each $w_{i,j} \in L$. Putting everything together, we have:

$$w = w_{1,1} \dots w_{1,k_1} w_{2,1} \dots w_{2,k_2} \dots w_{n,1} \dots w_{n,k_n}$$

  Since each component is in L, $w \in L^*$.

2.(b) We show that for any language L, $L \cdot L^* = L^* \cdot L$.

- $L \cdot L^* \subseteq L^* \cdot L$: let $w \in L \cdot L^*$. Then $w = \widehat{w} w_1 \dots w_n$ where $\widehat{w} \in L$ and $\forall i. w_i \in L$. Then since we have $\widehat{w} w_1 \dots w_{n-1} \in L^*$ and $w_n \in L$, we obtain $w \in L^* \cdot L$.
- The other direction is similar.

2.(c) We want to show that for every language L, $\{\varepsilon\} \cup L \cdot L^* = L^*$. Remember that by definition, $L^* = \bigcup_{n\in\mathbb{N}} L^n$, and $L^0 = \{\varepsilon\}$. Then:

$$\begin{aligned}
\{\varepsilon\} \cup L \cdot L^* &= L^0 \cup L \cdot (\bigcup_{n\in\mathbb{N}} L^n) \\
&= L^0 \cup \bigcup_{n\in\mathbb{N}} L \cdot L^n \\
&= L^0 \cup \bigcup_{n\in\mathbb{N}} L^{n+1} \\
&= L^0 \cup \bigcup_{n\geq 1} L^n \\
&= \bigcup_{n\in\mathbb{N}} L^n \\
&= L^*
\end{aligned}$$

2.(d) We show that for all languages L and K: $(L \cup K)^* = (L^* \cdot K^*)^*$ by double inclusion.

- Let $w \in (L \cup K)^*$. So, w can be decomposed as $w = w_1 \dots w_n$ where for every i, either $w_i \in L$ or $w_i \in K$.
  If $w_i \in L$, then we also have $w_i \in L^*$, and since $\varepsilon \in K^*$, we have $w_i = w_i \cdot \varepsilon \in L^* \cdot K^*$.
  If $w_i \in K$, by the same reasoning, $w_i \in L^* \cdot K^*$.
  Since all the components of w are in $L^* \cdot K^*$, then $w \in (L^* \cdot K^*)^*$.
- We show by induction on n that if $w \in (L^*K^*)^n$, then $w \in (L \cup K)^*$.
  - $n = 0$: OK since the only possible w is ε.
  - Let $w \in (L^*K^*)^{n+1}$ and assume by induction hypothesis that $(L^*K^*)^n \subseteq (L \cup K)^*$.
    We can decompose $w = w'w''$ with $w' \in L^*K^*$ and $w'' \in (L^*K^*)^n \subseteq (L \cup K)^*$. The word w' can be further decomposed as $w' = u_1 \dots u_k v_1 \dots v_\ell$, where each $u_i \in L$ and each $v_i \in K$. Moreover, since $w'' \in (L \cup K)^*$, we can decompose it as $w'' = x_1 \dots x_m$ where each x_i is either in L or in K.
    Putting all the pieces together, $w = u_1 \dots u_k v_1 \dots v_\ell x_1 \dots x_m$ where each component is either in L or in K. So $w \in (L \cup K)^*$.

  Since $(L^*K^*)^* = \bigcup_{n\in\mathbb{N}} (L^*K^*)^n$, this implies that $(L^*K^*)^* \subseteq (L \cup K)^*$.

Exercice 6: Let e and f be two regular expressions such that $[\![ef]\!] \subseteq [\![f]\!]$.

1a. Prove that for all words $w_1, \dots, w_n \in [\![e]\!]$, and all words $w' \in [\![f]\!]$, we have $w_1 \cdots w_n w' \in [\![f]\!]$ (by induction on $n \in \mathbb{N}$).

1b. Prove that $[\![e^* f]\!] \subseteq [\![f]\!]$.

2. Guess what can be stated if one assumes that $[\![fe]\!] \subseteq [\![f]\!]$ (instead of $[\![ef]\!] \subseteq [\![f]\!]$).

Solutions:

1a. By induction on n:

- For $n = 0$, if $w' \in [\![f]\!]$ then $w' \in [\![f]\!]$ (trivially).
- Let $w_1, \ldots, w_{n+1} \in [\![e]\!]$ and $w' \in [\![f]\!]$. Since $w_{n+1}w' \in [\![e]\!][\![f]\!] = [\![ef]\!]$, and we have $[\![ef]\!] \subseteq [\![f]\!]$, then $w_{n+1}w' \in [\![f]\!]$. Then, by induction hypothesis, we have $w_1 \cdots w_n(w_{n+1}w') \in [\![f]\!]$.

1b. We want to prove $[\![e^* f]\!] \subseteq [\![f]\!]$, i.e., $[\![e]\!]^*[\![f]\!] \subseteq [\![f]\!]$.
Let $w \in [\![e]\!]^*[\![f]\!]$, then w is of the form $w = w_1 \ldots w_n w'$ with $w_i \in [\![e]\!]$ for all i and $w' \in [\![f]\!]$. By question 1a, we deduce $w \in [\![f]\!]$.

2. If $[\![fe]\!] \subseteq [\![f]\!]$, then $[\![fe^*]\!] \subseteq [\![f]\!]$.