Finite Automata Theory and Formal Languages

TMV027/DIT321– LP4 2018

Lecture 9

Ana Bove

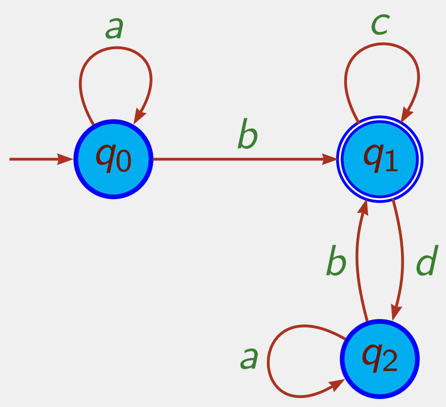
April 19th 2018

Recap: Regular Expressions

- Algebraic representation of (regular) languages;
- $R, S ::= \emptyset \mid \epsilon \mid a \mid R + S \mid RS \mid R^*$ ...
- ... representing the languages $\emptyset, \{\epsilon\}, \{a\}, \mathcal{L}(R) \cup \mathcal{L}(S), \mathcal{L}(R)\mathcal{L}(S)$ and $\mathcal{L}(R)^*$ respectively;
- Brief on algebraic laws for RE;
- How to transform a FA into a RE:
    - By eliminating states;
    - With a system of linear equations and Arden's lemma.

Example: Eliminating States

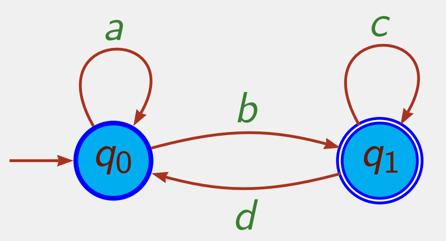
Consider the automaton D



By eliminating states the expression is

$$a^*b(c + da^*b)^*$$

Consider the automaton D'



By eliminating states the expression is

$$(a + bc^*d)^*bc^*$$

But intuitively these automata are equivalent...

Example: Linear Equation System

The linear equations corresponding to the automaton D' are

$$E_0 = aE_0 + bE_1 \qquad E_1 = \epsilon + cE_1 + dE_0$$

The resulting RE depends on the order we solve the system.

If we solve E_1 first we get $E_0 = (a + bc^*d)^*bc^*$.

If we solve E_0 first we get $E_0 = a^*b(c + da^*b)^*$.

It should be that $a^*b(c + da^*b)^* = (a + bc^*d)^*bc^*$! (see proof in slide 10)

Exercise: What RE do we obtain for the automaton D?

Overview of Today's Lecture

- Equivalence between FA and RE: from RE to FA;
- More on algebraic laws for regular expressions;
- Pumping Lemma for RL;
- Closure properties of RL.

Contributes to the following learning outcome:

- Explain and manipulate the diff. concepts in automata theory and formal lang;
- Have a clear understanding about the equivalence between (non-)deterministic finite automata and regular expressions;
- Understand the power and the limitations of regular lang and context-free lang;
- Prove properties of languages, grammars and automata with rigorously formal mathematical methods;
- Design automata, regular expressions and context-free grammars accepting or generating a certain language;
- Describe the language accepted by an automata or generated by a regular expression or a context-free grammar;
- Determine if a certain word belongs to a language;
- Differentiate and manipulate formal descriptions of lang, automata and grammars.

From Regular Expressions to Finite Automata

Proposition: *Every language defined by a RE is accepted by a FA.*

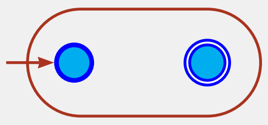
Proof: Let $\mathcal{L} = \mathcal{L}(R)$ for some RE R.

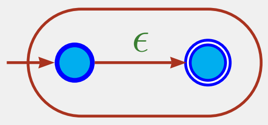
By induction on R we construct a ϵ-NFA E with only one final state and no arcs into the initial state or out of the final state.

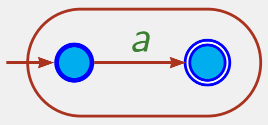
Moreover, E is such that $\mathcal{L} = \mathcal{L}(E)$.

Base cases are the RE $\emptyset$, ϵ and $a \in \Sigma$.

The corresponding ϵ-NFA accepting the languages $\emptyset$, $\{\epsilon\}$ and $\{a\}$ are:



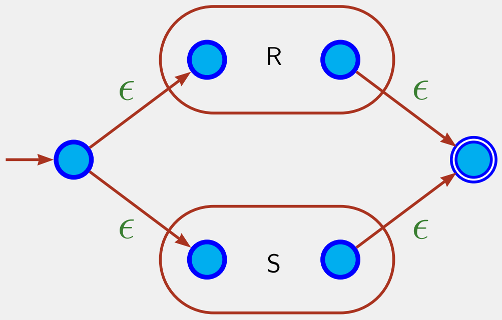


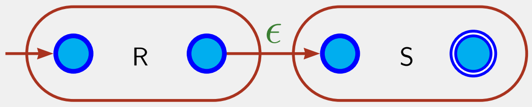


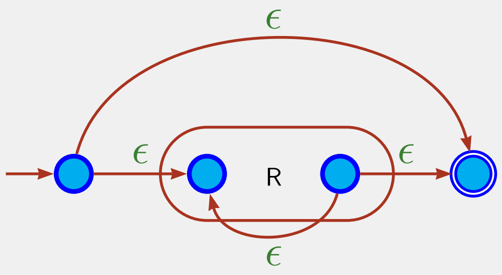
From RE to FA: Inductive Step (Cont.)

IH: Given the RE R and S there are ϵ-NFA with only one final state and no arcs into the initial state or out of the final state accepting $\mathcal{L}(R)$ and $\mathcal{L}(S)$.

We construct the ϵ-NFA for $R + S$, RS and R^* accepting $\mathcal{L}(R) \cup \mathcal{L}(S)$, $\mathcal{L}(R)\mathcal{L}(S)$ and $\mathcal{L}(R)^*$ respectively:

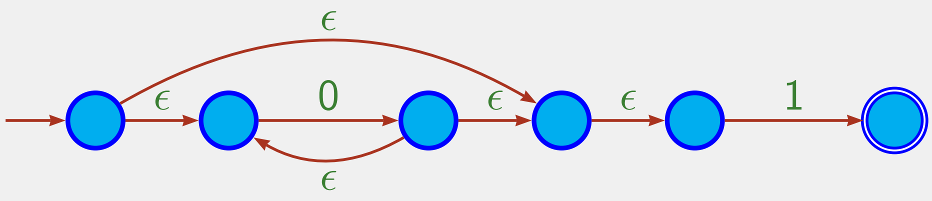




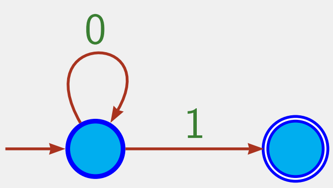


Example: From RE to FA

If we follow this method for the RE 0^*1 we obtain the ϵ-NFA



Compare it with the following DFA for the same language:



Recall: Algebraic Laws for Regular Expressions

The following equalities hold for any RE R, S and T:

Idempotent:	$R + R = R$	
Commutative:	$R + S = S + R$	In general, $RS \neq SR$
Associative:	$R + (S + T) = (R + S) + T$	$R(ST) = (RS)T$
Distributive:	$R(S + T) = RS + RT$	$(S + T)R = SR + TR$
Identity:	$R + \emptyset = \emptyset + R = R$	$R\epsilon = \epsilon R = R$
Annihilator:	$R\emptyset = \emptyset R = \emptyset$	
	$\emptyset^* = \epsilon^* = \epsilon$	
	$R^+ = RR^* = R^*R$	
	$R^* = (R^*)^* = R^*R^* = \epsilon + R^+$	

Note: Compare (some of) these laws with those for sets on slide 14 lecture 2.

Recall: More Algebraic Laws for Regular Expressions

Other useful laws to simplify regular expressions are:

- *Shifting rule:* $R(SR)^* = (RS)^*R$
- *Denesting rule:* $(R^*S)^*R^* = (R + S)^*$

  Note: By the shifting rule we also get $R^*(SR^*)^* = (R + S)^*$
- Variation of the denesting rule: $(R^*S)^* = \epsilon + (R + S)^*S$

Note: These rules are not always trivial to apply ... :-)

Example: Proving Equalities Using the Algebraic Laws

Example: The set of all words with no substring of more than two adjacent 0's is $(1+01+001)^*(\epsilon+0+00)$. Now,

$$
\begin{aligned}
&(1+01+001)^*(\epsilon+0+00) \\
&\quad = ((\epsilon+0)(\epsilon+0)1)^*(\epsilon+0)(\epsilon+0) && \text{by distributivity} \\
&\quad = (\epsilon+0)(\epsilon+0)(1(\epsilon+0)(\epsilon+0))^* && \text{by shifting} \\
&\quad = (\epsilon+0+00)(1+10+100)^* && \text{by distributivity}
\end{aligned}
$$

Then $(1+01+001)^*(\epsilon+0+00) = (\epsilon+0+00)(1+10+100)^*$

Example: A proof that $a^*b(c+da^*b)^* = (a+bc^*d)^*bc^*$ (recall slides 2 and 3):

$$
\begin{aligned}
a^*b(c+da^*b)^* &= a^*b(c^*da^*b)^*c^* && \text{by denesting } (R=c, S=da^*b) \\
a^*b(c^*da^*b)^*c^* &= (a^*bc^*d)^*a^*bc^* && \text{by shifting } (R=a^*b, S=c^*d) \\
(a^*bc^*d)^*a^*bc^* &= (a+bc^*d)^*bc^* && \text{by denesting } (R=a, S=bc^*d)
\end{aligned}
$$

Equality of Regular Expressions

Recall: RE are a way to denote languages.

Then, for RE R and S, $R = S$ actually means $\mathcal{L}(R) = \mathcal{L}(S)$.

Hence we can prove the equality of RE in the same way we prove the equality of languages!

Example: Let us show that $R^* = R^*R^*$. Let $\mathcal{L} = \mathcal{L}(R)$.

Then $\mathcal{L}(R^*) = \mathcal{L}(R)^* = \mathcal{L}^*$.

$\mathcal{L}^* \subseteq \mathcal{L}^*\mathcal{L}^*$ since $\epsilon \in \mathcal{L}^*$.

Conversely, if $\mathcal{L}^*\mathcal{L}^* \subseteq \mathcal{L}^*$ then $x = x_1x_2$ with $x_1 \in \mathcal{L}^*$ and $x_2 \in \mathcal{L}^*$.

If $x_1 = \epsilon$ or $x_2 = \epsilon$ then it is clear that $x \in \mathcal{L}^*$.

Otherwise $x_1 = u_1u_2 \ldots u_n$ with $u_i \in \mathcal{L}$ and $x_2 = v_1v_2 \ldots v_m$ with $v_j \in \mathcal{L}$.

Then $x = x_1x_2 = u_1u_2 \ldots u_nv_1v_2 \ldots v_m$ is in $\mathcal{L}^*$.

Proving Algebraic Laws for Regular Expressions

In general, given the RE R and S we can prove the law $R = S$ as follows:

1. Convert R and S into *concrete* RE C and D, respectively, by replacing each variable in the RE R and S by (different) concrete symbols.

   Example: $R(SR)^* = (RS)^*R$ can be converted into $a(ba)^* = (ab)^*a$.

2. Prove or disprove whether $\mathcal{L}(C) = \mathcal{L}(D)$. If $\mathcal{L}(C) = \mathcal{L}(D)$ then $R = S$ is a true law, otherwise it is not.

   Example: We can prove the shifting law by induction: $\forall n \in \mathbb{N}. a(ba)^n = (ab)^n a$.

Theorem: *The above procedure correctly identifies the true laws for RE.*

Proof: See theorems 3.14 and 3.13 in pages 121 and 120 respectively.

Example: Proving the Denesting Rule

We can state $(R^*S)^*R^* = (R+S)^*$ by proving $\mathcal{L}((a^*b)^*a^*) = \mathcal{L}((a+b)^*)$:

$\subseteq$: Let $x \in (a^*b)^*a^*$, then $x = vw$ with $v \in (a^*b)^*$ and $w \in a^*$.

By (structural) induction on v. If $v = \epsilon$ we are done.

Otherwise $v = av'$ or $v = bv'$.
In both cases $v' \in (a^*b)^*$ hence by IH $v'w \in (a+b)^*$ and so is vw.

$\supseteq$: Let $x \in (a+b)^*$.

By (structural) induction on x. If $x = \epsilon$ then we are done.

Otherwise $x = x'a$ or $x = x'b$ and $x' \in (a+b)^*$.

By IH $x' \in (a^*b)^*a^*$ and then $x' = vw$ with $v \in (a^*b)^*$ and $w \in a^*$.

If $x'a = v(wa) \in (a^*b)^*a^*$ since $v \in (a^*b)^*$ and $(wa) \in a^*$.
If $x'b = (v(wb))\epsilon \in (a^*b)^*a^*$ since $v(wb) \in (a^*b)^*$ and $\epsilon \in a^*$.

How to Identify Regular Languages?

We have seen that a language is regular iff there is a DFA that accepts the language.

Then we saw that DFA, NFA and ϵ-NFA are equivalent in the sense that we can convert between them.

Hence FA accept all and only the regular languages (RL).

Now we have seen how to convert between FA and RE.

Thus RE also define all and only the RL.

How to Prove that a Language is NOT Regular?

In a FA with n states, any path

$$q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} q_3 \xrightarrow{a_3} \dots \xrightarrow{a_{m-1}} q_m \xrightarrow{a_m} q_{m+1}$$

has a loop if $m \geqslant n$.

That is, we have $i < j$ such that $q_i = q_j$ in the path above.

This is an application of the *Pigeonhole Principle*.

How to Prove that a Language is NOT Regular?

Example: Let us prove that $\mathcal{L} = \{0^m 1^m | m \geqslant 0\}$ is NOT a RL.

Let us assume it is: then $\mathcal{L} = \mathcal{L}(A)$ for some FA A with n states, $n > 0$.

Let $k \geqslant n > 0$ and let $w = 0^k 1^k \in \mathcal{L}$.

Then there must be an accepting path $q_0 \xrightarrow{w} q_f \in F$.

Since $k \geqslant n$, there is a loop (pigeonhole principle) when reading the 0's.

Then $w = xyz$ with $|xy| = j \leqslant n$, $y \neq \epsilon$ and $z = 0^{k-j}1^k$ such that

$$q_0 \xrightarrow{x} q_l \xrightarrow{y} q_l \xrightarrow{z} q_f \in F$$

Observe that the following path is also an accepting path

$$q_0 \xrightarrow{x} q_l \xrightarrow{z} q_f \in F$$

However y must be of the form 0^i with $i > 0$ hence $xz = 0^{k-i}1^k \notin \mathcal{L}$.

This contradicts the fact that A accepts $\mathcal{L}$.

The Pumping Lemma for Regular Languages

Theorem: *Let $\mathcal{L}$ be a RL.*
Then, there exists a constant n—which depends on $\mathcal{L}$—such that for every string $w \in \mathcal{L}$ with $|w| \geqslant n$, it is possible to break w into 3 strings x, y and z such that $w = xyz$ and

1. $y \neq \epsilon$;
2. $|xy| \leqslant n$;
3. $\forall k \geqslant 0.\ xy^kz \in \mathcal{L}$.

Proof of the Pumping Lemma

Assume we have a FA A that accepts the language, then $\mathcal{L} = \mathcal{L}(A)$.

Let n be the number of states in A.

Then any path of length $m \geqslant n$ has a loop.

Let us consider $w = a_1 a_2 \ldots a_m \in \mathcal{L}$.

We have an accepting path and a loop such that

$$q_0 \xrightarrow{x} q_l \xrightarrow{y} q_l \xrightarrow{z} q_f \in F$$

with $w = xyz \in \mathcal{L}$, $y \neq \epsilon$, $|xy| \leqslant n$.

Then we also have

$$q_0 \xrightarrow{x} q_l \xrightarrow{y^k} q_l \xrightarrow{z} q_f \in F$$

for any k, that is, $\forall k \geqslant 0.\ xy^kz \in \mathcal{L}$.

Example: Application of the Pumping Lemma

We use the Pumping lemma to prove that $\mathcal{L} = \{0^m1^m | m \geqslant 0\}$ is not a RL.

We assume it is. Then the Pumping lemma applies.

Let n be the constant given by the lemma and let $w = 0^n1^n \in \mathcal{L}$, then $|w| \geqslant n$.

By the lemma we know that $w = xyz$ with $y \neq \epsilon$, $|xy| \leqslant n$ and $\forall k \geqslant 0.\ xy^kz \in \mathcal{L}$.

Since $y \neq \epsilon$ and $|xy| \leqslant n$, we know that $y = 0^i$ with $i \geqslant 1$.

However, we have a contradiction since $xy^kz \notin \mathcal{L}$ for $k \neq 1$ since it either has too few 0's ($k = 0$) or too many ($k > 1$).

Note: This is connected to the fact that a FA has *finite memory*!
If we could build a machine with infinitely many states it would be able to recognise the language.

Example: Application of the Pumping Lemma

Example: Let us prove that $\mathcal{L} = \{0^i 1^j | i \leqslant j\}$ is not a RL.

Let us assume it is, hence the Pumping lemma applies.

Let n be given by the Pumping lemma and let $w = 0^n 1^{n+1} \in \mathcal{L}$, hence $|w| \geqslant n$.

Then we know that $w = xyz$ with $y \neq \epsilon$, $|xy| \leqslant n$ and $\forall k \geqslant 0.\ xy^k z \in \mathcal{L}$.

Since $y \neq \epsilon$ and $|xy| \leqslant n$, we know that $y = 0^r$ with $r \geqslant 1$.

However, we have a contradiction since $xy^k z \notin \mathcal{L}$ for $k > 2$ since it will have more 0's than 1's.

(Even for $k = 2$ if $r > 1$.)

Example: What happens if we choose $w = 1^n \in \mathcal{L}$? Or $w = 0^{n/2} 1^n \in \mathcal{L}$?

Exercise: What about the languages $\{0^i 1^j \mid i \geqslant j\}$, $\{0^i 1^j \mid i > j\}$ and $\{0^i 1^j \mid i \neq j\}$?

Pumping Lemma is not a Sufficient Condition

By showing that the Pumping lemma does not apply to a certain language $\mathcal{L}$ we prove that $\mathcal{L}$ is not regular.

However, if the Pumping lemma *does* apply to $\mathcal{L}$, we *cannot* conclude whether $\mathcal{L}$ is regular or not!

Example: We know $\mathcal{L} = \{b^m c^m \mid m \geqslant 0\}$ is not regular.

Let us consider $\mathcal{L}' = a^+ \mathcal{L} \cup (b + c)^*$.

Using clousure properties (to come later) we can prove that $\mathcal{L}'$ is not regular.

However, the Pumping lemma does apply for $\mathcal{L}'$ with $n = 1$.

This shows the Pumping lemma is not a sufficient condition for a language to be regular.

Closure Properties for Regular Languages

Let $\mathcal{M}$ and $\mathcal{N}$ be RL. Then $\mathcal{M} = \mathcal{L}(R) = \mathcal{L}(D)$ and $\mathcal{N} = \mathcal{L}(S) = \mathcal{L}(F)$ for RE R and S, and DFA D and F.

We have seen that RL are closed under the following operations:

Union: $\mathcal{M} \cup \mathcal{N} = \mathcal{L}(R + S)$ or $\mathcal{M} \cup \mathcal{N} = \mathcal{L}(D \oplus F)$ (s.22, l.5);

Complement: $\overline{\mathcal{M}} = \mathcal{L}(\overline{D})$ (slide 24, lec. 5)

Intersection: $\mathcal{M} \cap \mathcal{N} = \overline{\overline{\mathcal{M}} \cup \overline{\mathcal{N}}}$ or $\mathcal{M} \cap \mathcal{N} = \mathcal{L}(D \otimes F)$ (s.21, l.5);

Difference: $\mathcal{M} - \mathcal{N} = \mathcal{M} \cap \overline{\mathcal{N}}$;

Concatenation: $\mathcal{M}\mathcal{N} = \mathcal{L}(RS)$;

Closure: $\mathcal{M}^* = \mathcal{L}(R^*)$.

More Closure Properties for Regular Languages

RL are also closed under the following operations:

Prefix: See additional exercise 3 on DFA. *Hint:* in D, make final all states in a path from the start state to final state.

Reversal: Recall that $\text{rev}(a_1 \ldots a_n) = a_n \ldots a_1$ and $\forall x. \text{rev}(\text{rev}(x)) = x$ (slides 15 & 17, lec. 4).

Closure under Prefix

Another way to prove that the language of prefixes of a RL is regular:

Define the function:

$$\begin{aligned}
&pre : RE \to RE \\
&pre(\emptyset) = \emptyset \\
&pre(\epsilon) = \epsilon \\
&pre(a) = \epsilon + a \\
&pre(R_1 + R_2) = pre(R_1) + pre(R_2) \\
&pre(R_1 R_2) = pre(R_1) + R_1 pre(R_2) \\
&pre(R^*) = R^* pre(R)
\end{aligned}$$

and prove that $\mathcal{L}(pre(R)) = \text{Prefix}(\mathcal{L}(R))$.

Then, if $\mathcal{L} = \mathcal{L}(R)$ for some RE R then $\text{Prefix}(\mathcal{L}) = \text{Prefix}(\mathcal{L}(R)) = \mathcal{L}(pre(R))$.

Closure under Reversal

We define the function:

$$\begin{array}{ll}
\_^{\mathsf{r}} : RE \to RE & \\
\emptyset^{\mathsf{r}} = \emptyset & (R_1 + R_2)^{\mathsf{r}} = R_1^{\mathsf{r}} + R_2^{\mathsf{r}} \\
\epsilon^{\mathsf{r}} = \epsilon & (R_1 R_2)^{\mathsf{r}} = R_2^{\mathsf{r}} R_1^{\mathsf{r}} \\
a^{\mathsf{r}} = a & (R^*)^{\mathsf{r}} = (R^{\mathsf{r}})^*
\end{array}$$

Theorem: *If $\mathcal{L}$ is regular so is $\mathcal{L}^{\mathsf{r}}$.*

Proof: (See theo. 4.11, pages 139–140).

Let R be a RE such that $\mathcal{L} = \mathcal{L}(R)$.
We need to prove by induction on R that $\mathcal{L}(R^{\mathsf{r}}) = (\mathcal{L}(R))^{\mathsf{r}}$.
Hence $\mathcal{L}^{\mathsf{r}} = (\mathcal{L}(R))^{\mathsf{r}} = \mathcal{L}(R^{\mathsf{r}})$ and $\mathcal{L}^{\mathsf{r}}$ is regular.

Example: The reverse of the language defined by $(0+1)^*0$ can be defined by $0(0+1)^*$.

Closure under Reversal

Another way to prove this result is by constructing a ϵ-NFA for $\mathcal{L}^{\mathrm{r}}$.

Proof: Let $N = (Q, \Sigma, \delta_N, q_0, F)$ be a NFA such that $\mathcal{L} = \mathcal{L}(N)$.

Define a ϵ-NFA $E = (Q \cup \{q\}, \Sigma, \delta_E, q, \{q_0\})$ with $q \notin Q$ and δ_E such that

$$\begin{array}{l} r \in \delta_E(s, a) \text{ iff } s \in \delta_N(r, a) \text{ for } r, s \in Q \\ r \in \delta_E(q, \epsilon) \text{ iff } r \in F \end{array}$$

Using Closure Properties

Example: Consider $\mathcal{L}_1$ and $\mathcal{L}_2$ such that $\mathcal{L}_1$ is regular, $\mathcal{L}_2$ is not regular but $\mathcal{L}_1 \cap \mathcal{L}_2$ is regular.

Is $\mathcal{L}_1 \cup \mathcal{L}_2$ is regular?

Let us assume that $\mathcal{L}_1 \cup \mathcal{L}_2$ is regular.

Then $(\mathcal{L}_1 \cup \mathcal{L}_2 - \mathcal{L}_1) \cup (\mathcal{L}_1 \cap \mathcal{L}_2)$ should also be regular because of the closure properties.

But this is actually $\mathcal{L}_2$ which is not regular!

We arrive to a contradiction.
Hence $\mathcal{L}_1 \cup \mathcal{L}_2$ cannot be regular.

Overview of Next Lecture

Sections 4.3–4.4:

- Decision properties for RL;
- Equivalence of RL;
- Minimisation of automata.

Overview of next Week

Mon 23	Tue 24	Wed 25	Thu 26	Fri 27
	Ex 10-12 EA RL.		**10-12 ES61 Individual help**	
Lec 13-15 HB3 RL.			**Lec 13-15 HB3** CFG.	
Ex 15-17 EA RL.		**15-17 EL41 Consultation**		

Assignment 4: RL.
Deadline: Sunday April 29th 23:59.