

# Regular Expressions

## A notation to describe “finite-automaton” patterns

---

E.g. Binary strings that “begin with a string of 0’s followed by a string of 1’s”.

Binary strings that “start and end with the same symbol”.

Regular expressions are just such a compact notation to describe these patterns, which are described respectively as  $0^* \cdot 1^*$  and  $0 + 1 + (0(0 + 1)^*0) + (1(0 + 1)^*1)$ .

### Regular expressions have many important applications in CS:

- Lexical analysis in compiler construction.
- Search facilities provided by text editors and databases; utilities such as `awk` and `grep` in Unix.
- Programming languages such as Perl and XML.

## Regular expressions and their denotations

---

Fix a  $\Sigma$ . We define simultaneously *regular expression*  $E$  and the *language denoted* by  $E$ , written  $L(E)$ , by induction over the following rules:

- The constants  $\epsilon$  and  $\emptyset$  are regular expressions;  
 $L(\epsilon) \stackrel{\text{def}}{=} \{\epsilon\}$  and  $L(\emptyset) \stackrel{\text{def}}{=} \emptyset$ .
- For  $a \in \Sigma$ ,  $a$  is a regular expression;  $L(a) \stackrel{\text{def}}{=} \{a\}$ .
- If  $E$  and  $F$  are regular expressions, then so are  $(E + F)$ ,  $(E \cdot F)$  and  $(E^*)$ ; we have

union	$L((E + F))$	$\stackrel{\text{def}}{=}$	$L(E) \cup L(F)$
concatenation	$L((E \cdot F))$	$\stackrel{\text{def}}{=}$	$L(E) \cdot L(F)$
star	$L((E^*))$	$\stackrel{\text{def}}{=}$	$(L(E))^*$

## Notations

---

$+$  is sometimes written  $\cup$  or  $|$ , and  $(E \cdot F)$  is sometimes simply written  $(EF)$ .

Parentheses may sometimes be omitted. We assume:

- (i) The regular expression operators have the following *order of precedence* (in decreasing order): star, concatenation, union.

E.g.  $01^*$  means  $0(1^*)$ , not  $(01)^*$ ;  $0 + 10$  means  $0 + (10)$ , not  $(0 + 1)0$ .

- (ii) Union and concatenation associate to the left i.e.  $E \cdot F \cdot G$  means  $(E \cdot F) \cdot G$ . (Since union and concatenation are associative, the choice of left or right association does not really matter.)

### Examples

(i)  $01^* + 1$  is formally  $((0 \cdot (1^*)) + 1)$ .

(ii)  $(0 + 1)01^*0$  is formally  $(((((0 + 1) \cdot 0) \cdot (1^*)) \cdot 0))$ .

## Examples: Languages over $\{0, 1\}$ denoted by regular expressions

---

1.  $0^*10^*$  denotes words that have exactly one 1.
2.  $(0 + 1)^*1(0 + 1)^*$  denote words that have at least one 1.
3.  $0(0 + 1)^*0 + 1(0 + 1)^*1 + 0 + 1$  denotes words that start and end with the same symbol.

We shall say that a word  $w$  *matches*  $E$  just in case  $w \in L(E)$ .

## Equivalence of regular expressions

---

We say that  $E$  and  $F$  are *equivalent*, written  $E \equiv F$ , just in case  $L(E) = L(F)$ .

Note that  $\equiv$  is an equivalence relation.

### Some identities

1. Associativity:  $(E + F) + G \equiv E + (F + G)$  and  $(EF)G \equiv E(FG)$ .
2. Commutativity:  $E + F \equiv F + E$
3.  $E\emptyset \equiv \emptyset$ .
4.  $\emptyset^* \equiv \{\epsilon\}$ .
5.  $E + \emptyset \equiv E \equiv \emptyset + E$  and  $E \cdot \epsilon \equiv E \equiv \epsilon \cdot E$ .
6. But *in general*  $E + \epsilon \not\equiv E$ , and  $E \cdot \emptyset \not\equiv E$ .

For which  $E$  do the equivalences hold?

### Example: Verify $(a + b)^* \equiv a^* (b a^*)^*$

**Proof.** Observe that  $L((a + b)^*)$  is the set of all strings over  $\{a, b\}$ , thus  $L(a^* (b a^*)^*) \subseteq L((a + b)^*)$ .

Note that any  $s \in L((a + b)^*)$  can be written *uniquely* as

$$a^{n_0} b a^{n_1} b \dots a^{n_{r-1}} b a^{n_r} \quad (1)$$

where  $a^n$  means  $\underbrace{a \dots a}_n$ , each  $n_i \geq 0$  and  $r \geq 0$ . ( $r$  is just the number of occurrences of  $b$  in  $s$ . E.g. in case  $s$  is  $b$ ,  $n_0 = n_1 = 0$ ; in case  $s$  is  $\epsilon$ ,  $n_0 = 0$ .)

Any string in  $L((b a^*)^*)$  has the shape  $\underbrace{(b a^{n_1}) \dots (b a^{n_r})}_r$ , where each  $n_i \geq 0$

and  $r \geq 0$ . It follows that any string in  $L(a^* (b a^*)^*)$  has the shape  $a^{n_0} b a^{n_1} \dots b a^{n_r}$  where each  $n_i \geq 0$  and  $r \geq 0$  i.e. of the shape (??).  $\square$

**Question:** Is there a finite set of (equivalence) axioms and rules such that  $(a + b)^* \equiv a^* (b a^*)^*$  (indeed any valid equivalence) is a theorem?

### Kozen's Axioms for the Algebra of Regular Expressions

- |   |   |
|---|---|
| 1. $E + (F + G) \equiv (E + F) + G$       | 7. $E(F + G) \equiv EF + EG$                        |
| 2. $E + F \equiv F + E$                   | 8. $(E + F)G \equiv EG + FG$                        |
| 3. $E + \emptyset \equiv E$               | 9. $\emptyset E \equiv E\emptyset \equiv \emptyset$ |
| 4. $E + E \equiv E$                       | 10. $\epsilon + EE^* \equiv E^*$                    |
| 5. $(EF)G \equiv E(FG)$                   | 11. $\epsilon + E^*E \equiv E^*$                    |
| 6. $\epsilon E \equiv E\epsilon \equiv E$ |   |

and two rules:

$$12. F + EG \leq G \Rightarrow E^*F \leq G$$

$$13. F + GE \leq G \Rightarrow FE^* \leq G$$

Note:  $E \leq F$  means  $L(E) \subseteq L(F)$ .

## (Sound and) Complete Axiomatization of Equivalence

---

**Soundness:** Each axiom is a valid equivalence between regular expressions, and each rule is **sound** (i.e. if the premise is a valid equivalence, so is the conclusion).

E.g. to say that rule (13) is sound is to say that for any  $E, F$  and  $G$ , if  $L(F + GE) \subseteq L(G)$ , then  $L(FE^*) \subseteq L(G)$ .

**Completeness:** Further the axiomatization is *complete* i.e.

**Kozen's Theorem.** All valid equivalences between regular expressions can be derived from Kozen's axioms and rules, using the laws of (in)equational logic i.e.

if  $E \leq F$  then  $E \oplus G \leq F \oplus G, G \oplus E \leq G \oplus F$  and  $E^* \leq F^*$   
where  $\oplus = +$  and  $\cdot$ .

## On proving Kozen's Theorem

---

**Soundness proof:** For an illustration, we prove (13).

Suppose  $L(F + GE) \subseteq L(G)$ . Any  $w \in L(FE^*)$  has the form  $fe_1 \cdots e_n$  where  $n \geq 0$ ,  $f \in L(F)$  and  $e_i \in L(E)$ . We prove that  $w \in L(G)$  by induction on  $n$ .

The base case of  $n = 0$  follows from  $L(F) \subseteq L(G)$ .

For the inductive case, we need to show that  $fe_1 \cdots e_{n+1} \in L(G)$ . Now  $fe_1 \cdots e_n \in L(FE^*)$ ; by the IH we have  $fe_1 \cdots e_n \in L(G)$ ; and so  $fe_1 \cdots e_n e_{n+1} \in L(GE)$  which is contained in  $L(G)$  by supposition.

**Completeness proof:** beyond the scope of this course. □

## Example: $(a + b)^* \equiv a^* (b a^*)^*$ revisited

---

We prove the harder direction “ $\leq$ ” using Kozen’s system.

Note that  $\epsilon, a, b \leq a^* (b a^*)^*$ . [Ex. Prove it using  $\epsilon \leq E^*$  !]

We have  $a (a^* (b a^*)^*) \equiv (a a^*) (b a^*)^* \leq a^* (b a^*)^*$  by (5) and (10). Similarly

$$b (a^* (b a^*)^*) \equiv (b a^*) (b a^*)^* \leq (b a^*)^* \equiv \epsilon (b a^*)^* \leq a^* (b a^*)^*.$$

The last two steps follow from (6) and (10). Because of the preceding, by (7), we have  $(a + b) + (a + b)(a^* (b a^*)^*) \leq a^* (b a^*)^*$ , and so, by rule (12) – taking  $E$  and  $F$  to be  $(a + b)$  – we have

$$(a + b)^* (a + b) \leq a^* (b a^*)^*.$$

Since  $\epsilon \leq a^* (b a^*)^*$ , by rule (10) we have  $(a + b)^* \leq a^* (b a^*)^*$  as desired.  $\square$

## Equivalence of regular expressions and finite automata

---

The equivalence of regular expressions and finite automata is a fundamental result in Automata Theory.

**Kleene’s Theorem.** Let  $L \subseteq \Sigma^*$ . The following are equivalent:

- (i)  $L$  is regular i.e. for some finite automaton  $M$ ,  $L = L(M)$ .
- (ii)  $L$  is denoted by some regular expression  $E$  i.e.  $L = L(E)$ .

## Proof of Kleene's Theorem: "(i) $\Rightarrow$ (ii)"

We show that there is a systematic way to transform a regular expression  $E$  to an equivalent NFA  $N_E$  – so that  $L(E) = L(N_E)$  – by recursion on the structure of  $E$ .

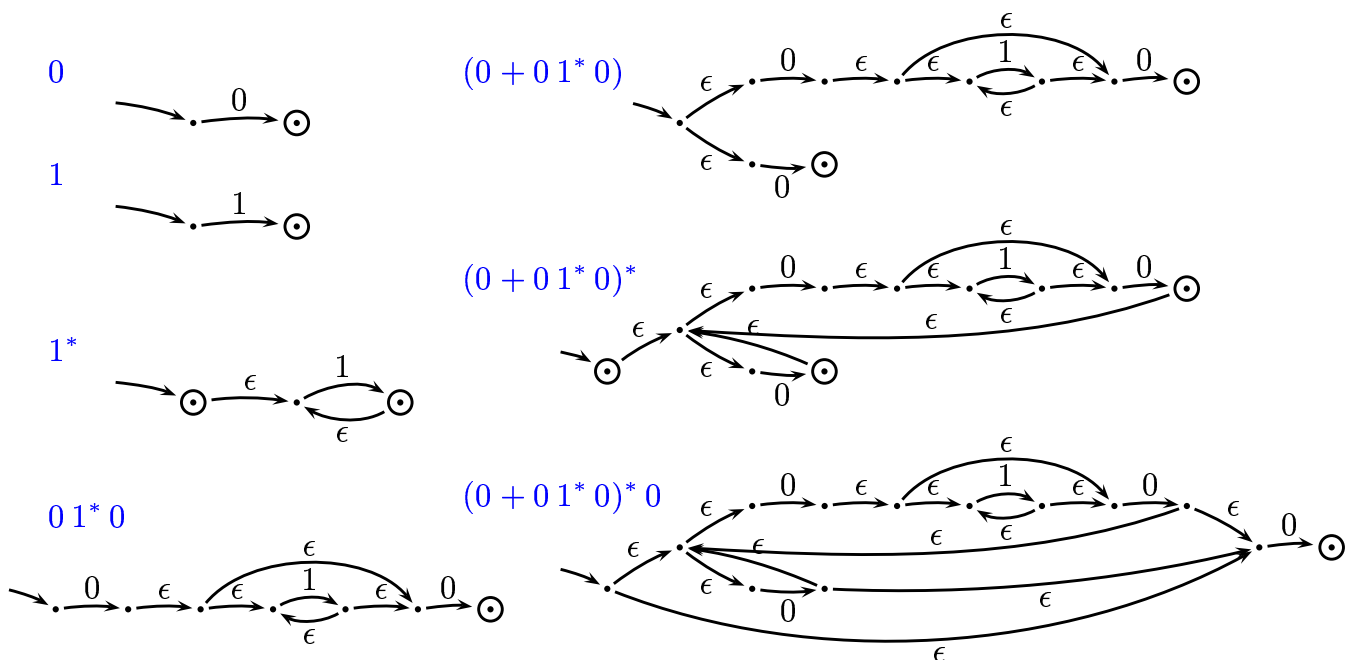
**Base cases:** For each of the three cases, namely  $E = \epsilon, \emptyset$  and  $a$  where  $a \in \Sigma$ , there is an NFA  $N_E$  that accepts  $L(E)$ .

**Inductive cases:** Take regular expressions  $E$  and  $F$ . Suppose  $N_E$  and  $N_F$  are NFAs that accept  $L(E)$  and  $L(F)$  respectively. We have proved that regular languages are closed under union, concatenation and star by constructing NFAs that accept  $L(N_E) \cup L(N_F)$ ,  $L(N_E) \cdot L(N_F)$  and  $(L(N_E))^*$  respectively. By definition, these NFAs are equivalent to  $E + F$ ,  $E \cdot F$  and  $E^*$  respectively.

□

## Example: Transforming regular expressions to NFAs

We construct the NFA that accepts  $(0 + 01^*0)^*0$ .



## Proof of Kleene's Theorem: "(ii) $\Rightarrow$ (i)"

Given an NFA  $M = (Q, \Sigma, \delta, q_0, F)$ , for  $X \subseteq Q$  and  $q, q' \in Q$ , we construct, by induction on the size of  $X$ , a regular expression

$$E_{q,q'}^X$$

whose denotation is the set of all strings  $w$  such that

there is a path from  $q$  to  $q'$  in  $M$  labelled by  $w$  (i.e.  $q \xrightarrow{w} q'$ ) such that all intermediate states along that path lie in  $X$ .

It suffices to prove:

**Lemma.** For any  $X \subseteq Q$ , for any  $q, q' \in Q$ , there is a regular expression  $E_{q,q'}^X$  satisfying  $L(E_{q,q'}^X) = \{ w \in \Sigma^* : q \xrightarrow{w} q' \text{ in } M \text{ with all intermediate states of seq. in } X \}$

We prove the Lemma by induction on the size of  $X$ .

**Basis:**  $X = \emptyset$ . Let  $a_1, \dots, a_k$  be all the symbols in  $\Sigma \cup \{ \epsilon \}$  such that  $q' \in \delta(q, a_i)$ . For  $q \neq q'$ , take

$$E_{q,q'}^{\emptyset} \stackrel{\text{def}}{=} \begin{cases} a_1 + \dots + a_k & \text{if } k \geq 1 \\ \emptyset & \text{if } k = 0 \end{cases}$$

and for  $q = q'$ , take

$$E_{q,q'}^{\emptyset} \stackrel{\text{def}}{=} \begin{cases} a_1 + \dots + a_k + \epsilon & \text{if } k \geq 1 \\ \epsilon & \text{if } k = 0 \end{cases}$$

**Inductive step:** For a nonempty  $X$ , choose an element  $r \in X$  - call it the *separating state*. Now any path from  $q$  to  $q'$  with all intermediate states in  $X$ , either

- (1) never visits  $r$ , or
- (2) visits  $r$  for the first time, followed by a finite number of loops from  $r$  back to itself without visiting  $r$  in between but staying in  $X$ , and finally followed by a path from  $r$  to  $q'$ .

Thus we take

$$E_{q,q'}^X \stackrel{\text{def}}{=} \underbrace{E_{q,q'}^{X-\{r\}}}_{(1)} + \underbrace{E_{q,r}^{X-\{r\}} \cdot (E_{r,r}^{X-\{r\}})^* \cdot E_{r,q'}^{X-\{r\}}}_{(2)}$$

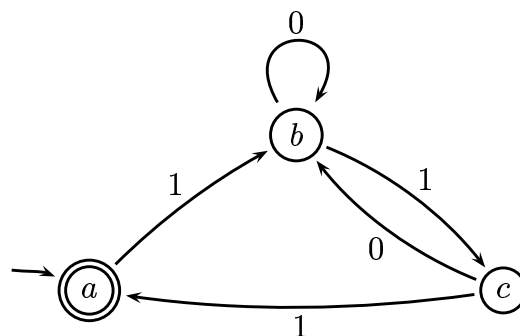
Finally the expression  $\sum_{f \in F} E_{q_0,f}^Q$  has denotation  $L(M)$ .  $\square$

**Hueristic:** it is best to choose a separating state  $r$  that disconnects the automaton as much as possible.

### Example: Transforming NFAs to regular expressions

Consider the NFA  $M = (\{a, b, c\}, \{0, 1\}, \delta, a, \{a\})$  where  $\delta$  is given by

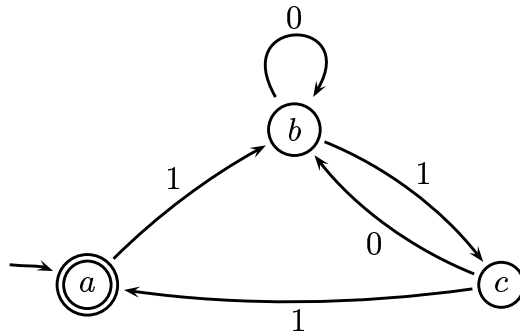
	0	1	$\epsilon$
a	$\emptyset$	$\{b\}$	$\emptyset$
b	$\{b\}$	$\{c\}$	$\emptyset$
c	$\{b\}$	$\{a\}$	$\emptyset$



We pick  $b$  as the separating state: the required reg. exp. is

$$E_{a,a}^{\{a,b,c\}} = E_{a,a}^{\{a,c\}} + E_{a,b}^{\{a,c\}} \cdot (E_{b,b}^{\{a,c\}})^* \cdot E_{b,a}^{\{a,c\}}.$$

By inspection  $E_{a,a}^{\{a,c\}} = \epsilon$ ,  $E_{a,b}^{\{a,c\}} = 1$  and  $E_{b,a}^{\{a,c\}} = 11$ .



Picking  $c$  as the separating state, we have

$$E_{b,b}^{\{a,c\}} = E_{b,b}^{\{a\}} + E_{b,c}^{\{a\}} \cdot (E_{c,c}^{\{a\}})^* \cdot E_{c,b}^{\{a\}}$$

where  $E_{b,b}^{\{a\}} = 0 + \epsilon$ ,  $E_{b,c}^{\{a\}} = 1$ ,  $E_{c,c}^{\{a\}} = \epsilon$  and  $E_{c,b}^{\{a\}} = 0 + 11$ .

Hence putting it all together we have

$$E_{a,a}^{\{a,b,c\}} = \epsilon + 1(0 + \epsilon + 1\epsilon^*(0 + 11))^* 11 \equiv \epsilon + 1(0 + 10 + 111)^* 11$$

i.e.  $L(M) = L(\epsilon + 1(0 + 10 + 111)^* 11)$ .

## Question

Is there a procedure (algorithm) that, given a string  $s$  and a regular expression  $E$ , will decide whether or not  $s$  matches  $E$ ?