Regular Expressions

Kamal Lodaya

BGVS Karnataka
Indian Institute of Science, Bengaluru

January 2024

Outline

Examples of Regular Expressions

Expressions built from a, b, ϵ, using operators $+$, $\cdot$, and $*$.

- $(a^*) \cdot b$
  "any number of a's followed by a single b."
- $(a+b)^* abb (a+b)^*$
  "contains abb as a subword."
- $(a+b)^* b (a+b)(a+b)$
  "3rd last letter is a b."
- $(b^* a b^* a)^* b^*$

Examples of Regular Expressions

Expressions built from a, b, ϵ, using operators $+$, $\cdot$, and $*$.

- $(a^*) \cdot b$
  "any number of a's followed by a single b."
- $(a+b)^* abb(a+b)^*$
  "contains abb as a subword."
- $(a+b)^* b(a+b)(a+b)$
  "3rd last letter is a b."
- $(b^* ab^* a)^* b^*$
  "Even number of a's."

Examples of Regular Expressions

Expressions built from a, b, ϵ, using operators $+$, $\cdot$, and $*$.

- $(a^*) \cdot b$
  "any number of a's followed by a single b."
- $(a+b)^* abb(a+b)^*$
  "contains abb as a subword."
- $(a+b)^* b(a+b)(a+b)$
  "3rd last letter is a b."
- $(b^* ab^* a)^* b^*$
  "Even number of a's."
- Ex. Give regexp for "Every 4-bit block of the form $w[4i, 4i+1, 4i+2, 4i+3]$ has even parity."

Examples of Regular Expressions

Expressions built from a, b, ϵ, using operators $+$, $\cdot$, and $*$.

- $(a^*) \cdot b$
  "any number of a's followed by a single b."
- $(a+b)^* abb(a+b)^*$
  "contains abb as a subword."
- $(a+b)^* b(a+b)(a+b)$
  "3rd last letter is a b."
- $(b^* ab^* a)^* b^*$
  "Even number of a's."
- Ex. Give regexp for "Every 4-bit block of the form $w[4i, 4i+1, 4i+2, 4i+3]$ has even parity."
  $(0000 + 0011 + \cdots + 1111)^*(\epsilon + 0 + 1 + \cdots + 111)$

Formal definitions

- Syntax of regular expressions over an alphabet A:

$$r ::= \emptyset \mid a \mid r + r \mid r \cdot r \mid r^*$$

  where $a \in A$.

- Semantics: associate a language $L(r) \subseteq A^*$ with regexp r.

$$\begin{array}{lcl} L(\emptyset) & = & \{\} \\ L(a) & = & \{a\} \\ L(r + r') & = & L(r) \cup L(r') \\ L(r \cdot r') & = & L(r) \cdot L(r') \\ L(r^*) & = & L(r)^* = \bigcup_{i=0}^{\infty} L(r)^i \ (\mathit{finite?infinite?}) \end{array}$$

Formal definitions

- Syntax of regular expressions over an alphabet A:

$$r ::= \emptyset \mid a \mid r + r \mid r \cdot r \mid r^*$$

  where $a \in A$.

- Semantics: associate a language $L(r) \subseteq A^*$ with regexp r.

$$\begin{array}{lcl} L(\emptyset) & = & \{\} \\ L(a) & = & \{a\} \\ L(r + r') & = & L(r) \cup L(r') \\ L(r \cdot r') & = & L(r) \cdot L(r') \\ L(r^*) & = & L(r)^* = \bigcup_{i=0}^{\infty} L(r)^i \ (finite?infinite?) \end{array}$$

- Question: What is L^0 ? By definition, $L^0 = \{\epsilon\}$

Formal definitions

- Syntax of regular expressions over an alphabet A:

$$r ::= \emptyset \mid a \mid r + r \mid r \cdot r \mid r^*$$

  where $a \in A$.

- Semantics: associate a language $L(r) \subseteq A^*$ with regexp r.

$$\begin{array}{lcl} L(\emptyset) & = & \{\} \\ L(a) & = & \{a\} \\ L(r + r') & = & L(r) \cup L(r') \\ L(r \cdot r') & = & L(r) \cdot L(r') \\ L(r^*) & = & L(r)^* = \bigcup_{i=0}^{\infty} L(r)^i \ (finite?infinite?) \end{array}$$

- Question: What is L^0 ? By definition, $L^0 = \{\epsilon\}$
  Question: Do we need ϵ in syntax?

Formal definitions

- Syntax of regular expressions over an alphabet A:

$$r ::= \emptyset \mid a \mid r + r \mid r \cdot r \mid r^*$$

  where $a \in A$.

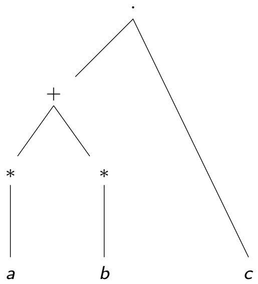
- Semantics: associate a language $L(r) \subseteq A^*$ with regexp r.

$$\begin{array}{lcl} L(\emptyset) & = & \{\} \\ L(a) & = & \{a\} \\ L(r + r') & = & L(r) \cup L(r') \\ L(r \cdot r') & = & L(r) \cdot L(r') \\ L(r^*) & = & L(r)^* = \bigcup_{i=0}^{\infty} L(r)^i \ (finite?infinite?) \end{array}$$

- Question: What is L^0 ? By definition, $L^0 = \{\epsilon\}$
  Question: Do we need ϵ in syntax? No. $L(\emptyset^*) = \{\epsilon\}$.

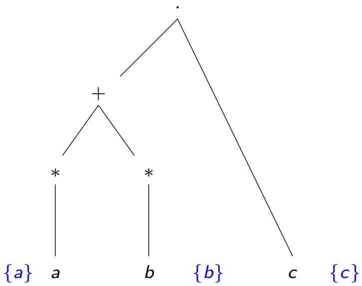
Example: Semantics of regexp

$(a^* + b^*) \cdot c$



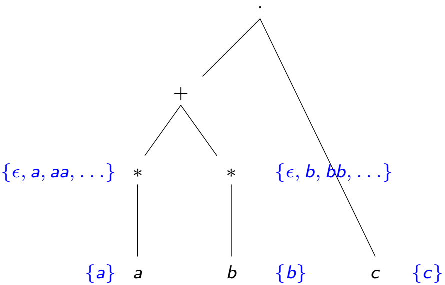
Example: Semantics of regexp

$(a^* + b^*) \cdot c$



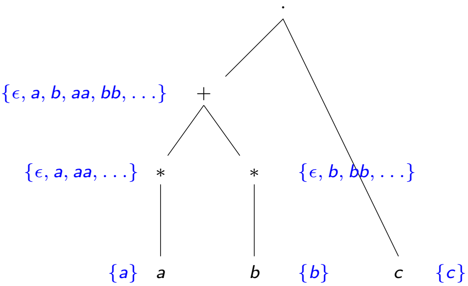
Example: Semantics of regexp

$(a^* + b^*) \cdot c$



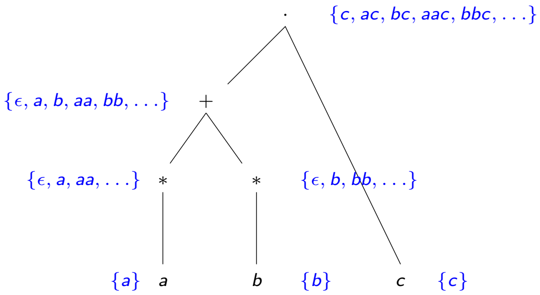
Example: Semantics of regexp

$(a^* + b^*) \cdot c$

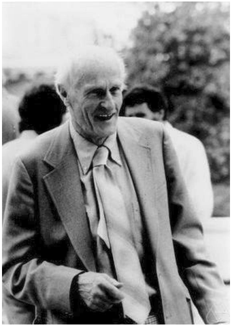


Example: Semantics of regexp

$$(a^* + b^*) \cdot c$$



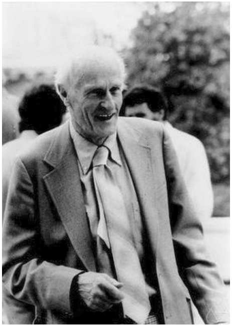
Kleene's Theorem (1956): RE = DFA



Class of languages defined by regular expressions coincides with regular languages.

Direction RE $\rightarrow$ NFA: Use closure properties of regular languages.
Idea: Construct (finite) NFA from given RE and $\rightarrow$ (finite) DFA using subset construction. Recall:

Kleene's Theorem (1956): RE = DFA

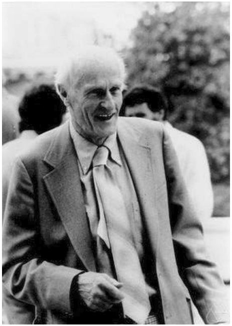


Class of languages defined by regular expressions coincides with regular languages.

Direction RE $\rightarrow$ NFA: Use closure properties of regular languages.
Idea: Construct (finite) NFA from given RE and $\rightarrow$ (finite) DFA using subset construction. Recall:

- $L \cup M = \{u \mid u \in L \text{ or } u \in M\}$ (NFA size?)

Kleene's Theorem (1956): RE = DFA



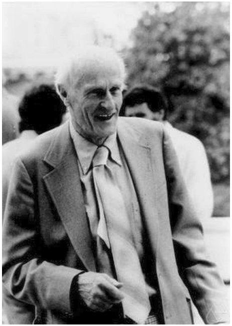
Class of languages defined by regular expressions coincides with regular languages.

Direction RE $\rightarrow$ NFA: Use closure properties of regular languages.
Idea: Construct (finite) NFA from given RE and $\rightarrow$ (finite) DFA using subset construction. Recall:

- $L \cup M = \{u \mid u \in L \text{ or } u \in M\}$ (NFA size?)
- $L \cdot M = \{u \cdot v \mid u \in L,\ v \in M\}$ (NFA size?)

Kleene's Theorem (1956): RE = DFA

Class of languages defined by regular expressions coincides with regular languages.



Direction RE $\rightarrow$ NFA: Use closure properties of regular languages.
Idea: Construct (finite) NFA from given RE and $\rightarrow$ (finite) DFA using subset construction. Recall:

- $L \cup M = \{u \mid u \in L \text{ or } u \in M\}$ (NFA size?)
- $L \cdot M = \{u \cdot v \mid u \in L,\ v \in M\}$ (NFA size?)
- $L^n = L \cdot L \cdot \dots \cdot L$ (n times), where $L^0 = \{\epsilon\}$ by definition
- $L^* = \bigcup_{i=0}^{\infty} L^i = L^0 \cup L^1 \cup L^2 \cup L^3 \dots$ (infinite union of any finite number of concatenations of L, but single NFA)

Glushkov construction for NFA $\mathcal{A}$ recognizing L

- Make all initial states non-initial and add new initial state e. For every transition (s_i, a, t) of $\mathcal{A}$ add transition (e, a, t). (Self-loop (s_i, a, s_i) of $\mathcal{A}$ results in (e, a, s_i).) If $a^* \subseteq L(s_i)$, the self-loop at s_i makes $a^+ \subseteq L(e)$. This is a single-source NFA for $L \setminus \{\epsilon\}$.
- Make all final states non-final and add new final state g. For every transition (t, b, f_i) of $\mathcal{A}$ add transition (t, b, g). (Self-loop (f_i, b, f_i) of $\mathcal{A}$ results in (f_i, b, g), making f_i nondeterministic, still $b^+ \subseteq L(f_i)$.)
- If s_i were initial and final in $\mathcal{A}$ and had self-loop (s_i, b, s_i) accepting $b^* \subseteq L(s_i)$, in the first step we get (e, b, s_i) with $b^+ \subseteq L(e)$. In the second step we get (e, b, g) accepting b, $b^+ \subseteq L(s_i)$, hence $b^+ \subseteq L(e)$ using edge for b and path through s_i. This is a single-source single-sink NFA for $L \setminus \{\epsilon\}$.

RE $\to$ NFA: closure under star

- Fuse e and g in the Glushkov construction into a single start and final state h. (Edges (e, c, g) become self-loops (h, c, h).)

  Paths of L cycling through h for the first and last letter accept strings in $L^+ = L^* \setminus \{\epsilon\}$ which move out of an initial state. Self-loops simulate remaining paths, so all of L^* is accepted.

  Do any new strings outside L^* get accepted? No. Every non-singleton cycle from h to h not visiting h inbetween has to go through the $\mathcal{A}$-states simulating a path of $L \setminus \{\epsilon\}$, giving cyclic paths from h to h in L^+. The self-loops at h correspond to three possibilities from $\mathcal{A}$: c, c^+ and c^*. But for all three cases $c^* = (c^+)^* = (c^*)^* \subseteq L^*$.

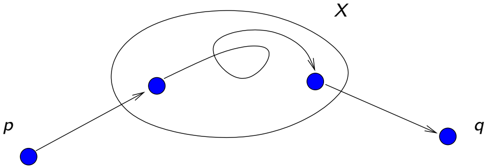
Direction NFA $\to$ RE: If F is empty, desired expression is $\emptyset$.
(Question: If there are no paths from S to F, should return $\emptyset$.)

Otherwise induction on number of states in given NFA.
Base case is one state.

NFA → RE: McNaughton-Yamada construction

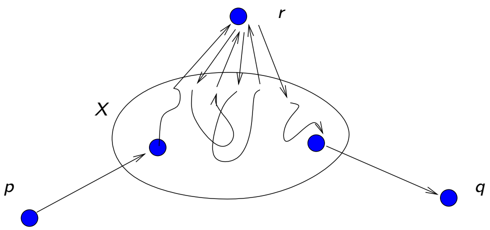
- Let $\mathcal{A} = (Q, S, \Delta, F)$ be given nonempty NFA.
- Define $L_{pq} = \{w \in A^* \mid q \in \widehat{\Delta}(p, w)\}$.
- Then $L(\mathcal{A}) = \bigcup_{s \in S} \bigcup_{f \in F} L_{sf}$.

- By induction on $|X|$, for $X \subseteq Q$ define $L_{pq}^X = \{w \in A^* \mid q \in \widehat{\Delta}(p, w), \text{ via a path that stays in } X \text{ for intermediate states}\}$



- Then $L(\mathcal{A}) = \bigcup_{s \in S} \bigcup_{f \in F} L_{sf}^Q$.

NFA $\rightarrow$ RE: McNaughton-Yamada construction



Induction:

$$L_{pq}^{X \cup \{r\}} = L_{pq}^{X} + L_{pr}^{X} \cdot (L_{rr}^{X})^{*} \cdot L_{rq}^{X}$$

NFA → RE: McNaughton-Yamada construction (2)

Method:

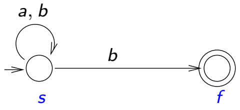
- Begin with L_{sf}^{Q} for each $s \in S$, $f \in F$.
- Simplify by using terms with strictly smaller X's:

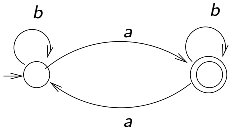
$$L_{pq}^{X \cup \{r\}} = L_{pq}^{X} + L_{pr}^{X} \cdot (L_{rr}^{X})^{*} \cdot L_{rq}^{X}.$$

- For base of the induction, observe that

$$L_{pq}^{\{\}} = \begin{cases} \{a \mid q \in \Delta(p,a)\} & \text{if} \quad p \neq q \\ \{a \mid q \in \Delta(p,a)\} \cup \{\epsilon\} & \text{if} \quad p = q. \end{cases}$$

- Exercise: convert NFA/DFA's below to RE's:





NFA → RE: McNaughton-Yamada construction (2)

Method:

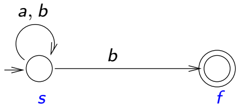
- Begin with L_{sf}^{Q} for each $s \in S$, $f \in F$.
- Simplify by using terms with strictly smaller X's:

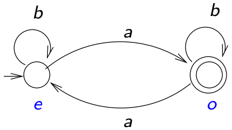
$$L_{pq}^{X \cup \{r\}} = L_{pq}^{X} + L_{pr}^{X} \cdot (L_{rr}^{X})^* \cdot L_{rq}^{X}.$$

- For base of the induction, observe that

$$L_{pq}^{\{\}} = \begin{cases} \{a \mid q \in \Delta(p, a)\} & \text{if} \quad p \neq q \\ \{a \mid q \in \Delta(p, a)\} \cup \{\epsilon\} & \text{if} \quad p = q. \end{cases}$$

- Exercise: convert NFA/DFA's below to RE's:





Corollary: checking language of NFA is nonempty

Define $L_{pq}^{X} = \{w \in A^* \mid q \in \widehat{\Delta}(p, w),\ \text{via path using intermediate states in } X \text{ at most once}\}$

- $L(\mathcal{A}) \neq \emptyset \iff (S \cap F \neq \emptyset) \vee \bigvee_{s \in S} \bigvee_{f \in F} (L_{sf}^{Q \setminus \{s,f\}} \neq \emptyset)$.
- Induction (X, Y, $\{p, q, r\}$ all disjoint):

$$L_{pq}^{X \cup Y \cup \{r\}} \neq \emptyset \iff (L_{pq}^{X \cup Y} \neq \emptyset) \vee \bigvee_{r \in Q} ((L_{pr}^{X} \neq \emptyset) \wedge (L_{rq}^{Y} \neq \emptyset))$$

- Base:

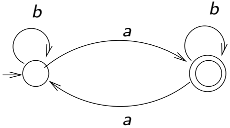
$$L_{pq}^{\{\}} \neq \emptyset \iff \begin{cases} \bigvee_{a \in A} (q \in \Delta(p, a)) & \text{if} \quad p \neq q \\ true & \text{if} \quad p = q. \end{cases}$$

Chomsky-Miller: Seeing NFA as system of equations

- Aim: to construct a regexp for

$$L_q = \{w \in A^* \mid \widehat{\Delta}(q, w) \cap F \neq \emptyset\}.$$

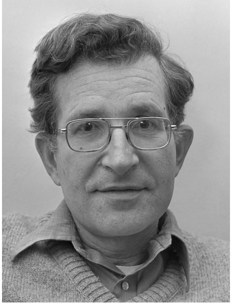
- Note that $L(\mathcal{A}) = \bigcup_{s \in S} L_s$. Example:



- Set up right-linear equations for L_q's:

$$\begin{aligned} x_e &= b \cdot x_e + a \cdot x_o \\ x_o &= a \cdot x_e + b \cdot x_o + \epsilon. \end{aligned}$$

- Solution is a RE for each x, such that languages of LHS and RHS coincide.

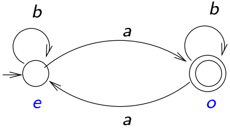


Chomsky-Miller: Seeing NFA as system of equations

- Aim: to construct a regexp for

$$L_q = \{w \in A^* \mid \widehat{\Delta}(q, w) \cap F \neq \emptyset\}.$$

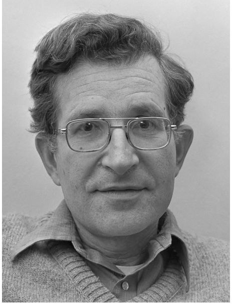
- Note that $L(\mathcal{A}) = \bigcup_{s \in S} L_s$. Example:



- Set up right-linear equations for L_q's:

$$\begin{aligned} x_e &= b \cdot x_e + a \cdot x_o \\ x_o &= a \cdot x_e + b \cdot x_o + \epsilon. \end{aligned}$$

- Solution is a RE for each x, such that languages of LHS and RHS coincide.



Conway: Solutions to a system of equations

- L_q's are a solution to the system of equations.
- In general there could be many solutions to equations.
- Consider $x = A^* x$ (Here A is the alphabet). What are the solutions to this equation?

- In the case of right-linear equations arising out of automata, L_q's can be seen to be the unique solution to the equations.

Conway: Least solution to a system of equations

- Equations arising from our automaton can be viewed as:

$$\begin{bmatrix} x_e \\ x_o \end{bmatrix} = \begin{bmatrix} b & a \\ a & b \end{bmatrix} \begin{bmatrix} x_e \\ x_o \end{bmatrix} + \begin{bmatrix} \emptyset \\ \epsilon \end{bmatrix}$$

- System of right-linear equations over regular expressions have the general form:

$$X = EX + F$$

  where X is a column vector of n variables, E is an $n \times n$ matrix of regular expressions, and F is a column vector of n regular expressions.

- Claim: The column vector E^*F represents the least solution to the equations above. [See Kozen, Supplementary Lecture A].
- Definition of E^* when E is a 2×2 matrix:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^* = \begin{bmatrix} (a + bd^*c)^* & (a + bd^*c)^*bd^* \\ (d + ca^*b)^*ca^* & (d + ca^*b)^* \end{bmatrix}$$

Schützenberger: formal power series semiring

A semiring is a set with binary associative operation $\cdot$ and binary commutative and associative operation $+$ and their units 1 and 0, with the first operation distributing over the second.

Examples: A^* with concatenation, union; $\mathbb{N}, \mathbb{R}^{\geq 0}$ with $\times, +$.

A weight function g maps a string to $\mathbb{N}$ such that every non-unit maps to a positive number and $g(u \cdot v) = g(u) + g(v)$. Thus $g(\epsilon) = 0$. More generally weights can be from a semiring $\mathbb{K}$.

A (proper) formal $\mathbb{K}$-power series over A has formal expressions representing weight functions $A^* \to \mathbb{K}$ (mapping ϵ to 0, or the special series ϵ with value 1 at ϵ and 0 elsewhere).
Proper power series allow infinite sums $t^* = \Sigma_{i\in\mathbb{N}} t^i$ (with $t^0 = \epsilon$).

Rational expressions (see Sakarovitch's chapter in HWA)

Syntax of $\mathbb{K}$-rational expresions over A, where $a \in A$ and $k \in \mathbb{K}$:

$$r ::= 0 \mid \epsilon \mid a \mid kr \mid rk \mid r + r \mid r \cdot r \mid r^*$$

Technicalities: $(a^* + (-1)b^*)^*$ is a valid $\mathbb{Z}$-rational expression, $(a^* + b^*)^*$ is not, since the inner expression has value 2 at ϵ, which is not proper and its star is undefined.

Semantics: associate a proper power series with expression r.

0 is the zero series with value 0 for all words w. $0^* = \epsilon = \epsilon^*$.
a has value 1 at word a and zero otherwise.
kr and rk left- and right-multiply the value at every word w by k.
We assume $s \cdot t$ maps w to $s(w) \cdot t(w)$ (more possibilities in HWA).

Proposition: $\mathbb{K}$-weighted rational expressions form a semiring.

Weighted automata

A $\mathbb{K}$-weighted automaton $\mathcal{A}$ with states Q is a square matrix E of dimension $|Q|$ with entries from a $\mathbb{K}$-power series, with initial row vector I and final column vector F.
Example: $\mathbb{B}$-weighted automaton over A.

The label of a computation $w_1 \dots w_n$ from p to q is the product $I_p w_1 \dots w_n F_q$. (A Glushkov automaton is one where I has a single non-zero coordinate with unit value and this unique initial state is not the target of any transition with non-zero label.)

The behaviour of an automaton is a power series representing labels of its computations.

Conway's theorem: The behaviour of an automaton equals $I \cdot E^* \cdot F$.

Kleene-Schützenberger theorem: A formal $\mathbb{K}$-power series is rational $\iff$ behaviour of a finite $\mathbb{K}$-weighted automaton.