COMP30026 Models of Computation

Lecture 11: Regular Expressions

Mak Nazecic-Andrlon and William Umboh

Semester 2, 2024

Regular Expressions

Compact notation for describing regular languages.

Similar to “regexes” in JavaScript or Python.

Example: $(0 \cup 1)(0 \cup 1)((0 \cup 1)(0 \cup 1))^*$ describes the strings whose lengths are a positive multiple of 2.

In Python: `r"(0|1)(0|1)((0|1)(0|1))*"`

Formal Syntax

The regular expressions over an alphabet $\Sigma = \{a_1, \ldots, a_n\}$ are given by the grammar

$$\begin{array}{lllllllllll} regexp & \rightarrow & a_1 & \mid & \cdots & \mid & a_n & \mid & \epsilon & \mid & \emptyset \\ & \mid & regexp \cup regexp & \mid & (regexp \circ regexp) & \mid & (regexp^*) \end{array}$$

Semantics:

$$\begin{array}{lcl} L(a) & = & \{a\} \\ L(\epsilon) & = & \{\epsilon\} \\ L(\emptyset) & = & \emptyset \\ L(R_1 \cup R_2) & = & L(R_1) \cup L(R_2) \\ L(R_1 \; R_2) & = & L(R_1) \circ L(R_2) \\ L(R^*) & = & L(R)^* \end{array}$$

Notational Conveniences

Can omit $\circ$ and sometimes parentheses.

Binding precedence: star $>$ concatenation $>$ union.

Examples:

1. ab means $(a \circ b)$
2. ab^* means $(a \circ (b^*))$.
3. $a \cup bc^*$ means $(a \cup (b \circ (c^*)))$.

Regular Expressions – Examples

$$
\begin{array}{rcl}
\epsilon & : & \{\epsilon\} \\
1 & : & \{1\} \\
110 & : & \{110\} \\
((0 \cup 1)(0 \cup 1))^* & : & \text{all binary strings of even length} \\
(0 \cup \epsilon)(\epsilon \cup 1) & : & \{\epsilon, 0, 1, 01\} \\
1^* & : & \text{all finite sequences of 1s} \\
\epsilon \cup 1 \cup (\epsilon \cup 1)^*(\epsilon \cup 1) & : & \text{all finite sequences of 1s} \\
(1^*0^*)^* & : & ?
\end{array}
$$

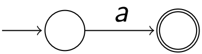
Regular Expressions to NFAs

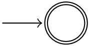
Theorem

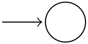
A language is regular iff it can be described by a regular expression.

Proof idea ($\Leftarrow$): Construct NFA from regular expression R. Use structural induction.

Base cases: $R = a \in \Sigma$, $R = \epsilon$, or $R = \emptyset$.

If $R = a$: Construct 

If $R = \epsilon$: Construct 

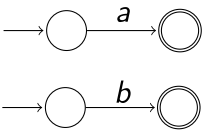
If $R = \emptyset$: Construct 

Inductive step: $R = R_1 \cup R_2$, $R = R_1 \circ R_2$, or $R = R_1^*$. Use the constructions for closure under regular operations.

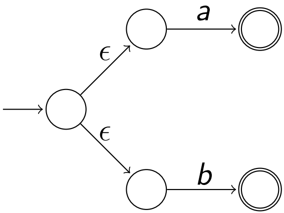
Regular Expressions to NFAs: Example

Convert $(\mathtt{a} \cup \mathtt{b})^*\mathtt{bc}$ to an NFA.

Start from innermost expressions and work out:

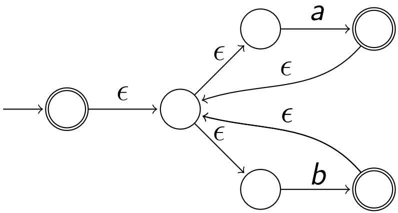


So an NFA for $a \cup b$ is:

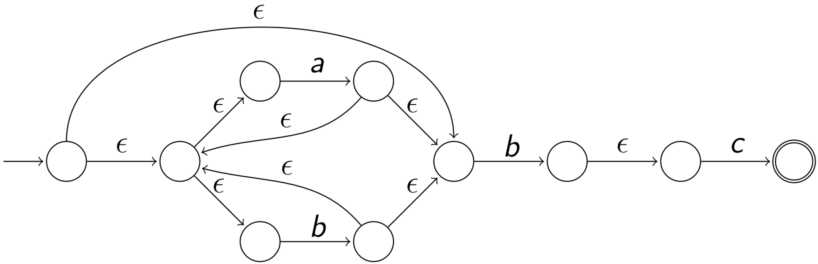


NFAs from Regular Expressions

Use star construction to get NFA for $(\mathtt{a} \cup \mathtt{b})^*$:



Finally, for $(\mathtt{a} \cup \mathtt{b})^*\mathtt{bc}$, we get:

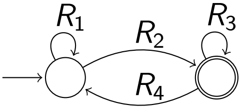


NFAs to Regular Expressions

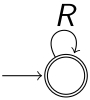
Proof idea (⇒): Reverse the construction; convert small pieces of NFA into matching regular expressions.

Represent using generalised NFAs (GNFAs), which allow labeling arrows with regular expressions.

The process produces either



or



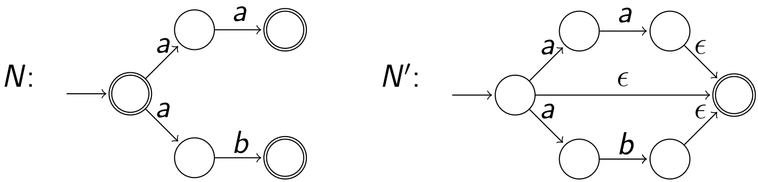
We get $(R_1 \cup R_2 R_3^* R_4)^* R_2 R_3^*$ in the first case; R^* in the second.

Note: some Rs may be ϵ or $\emptyset$.

Prove correctness by induction on number of states.

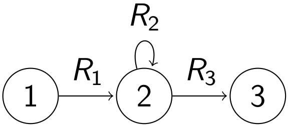
NFAs to Regular Expressions: Sketch

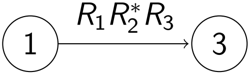
First, make sure there is only one accept state. Construction:



Next, we eliminate states that are neither start nor accept states.

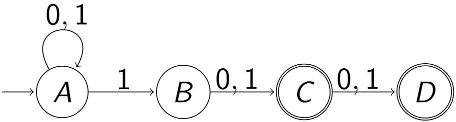
NFAs to Regular Expressions: Sketch

Replace 

with 

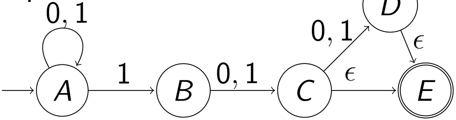
In general: If there are m incoming and n outgoing arrows, replace them with mn bypassing arrows.

Let us illustrate the process on this example:

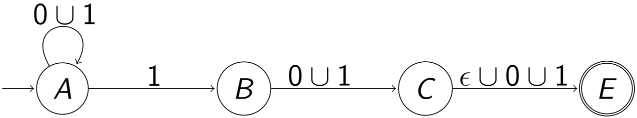


State Elimination Example

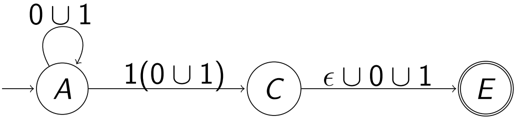
Create a single accept state:



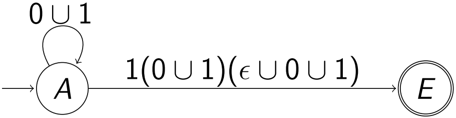
Eliminate D (and use regular expressions with all arcs):



Now eliminate B:

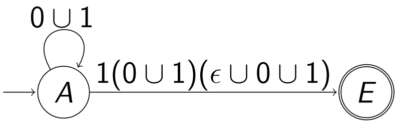


and then C:

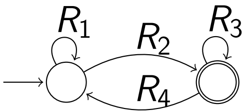


State Elimination Example

Note that



is



with

- $R_1 = 0 \cup 1$
- $R_2 = 1(0 \cup 1)(\epsilon \cup 0 \cup 1)$
- $R_3 = R_4 = \emptyset$

Hence the instance of the general "recipe" $(R_1 \cup R_2 R_3^* R_4)^* R_2 R_3^*$ is

$$(0 \cup 1)^* 1(0 \cup 1)(\epsilon \cup 0 \cup 1)$$

Full proofs in Sipser.

Some Useful Laws for Regular Expressions

$A \cup A = A$

$A \cup B = B \cup A$

$(A \cup B) \cup C = A \cup (B \cup C) = A \cup B \cup C$

$(A \circ B) \circ C = A \circ (B \circ C) = A \circ B \circ C$

$\emptyset \cup A = A \cup \emptyset = A$

$\epsilon \circ A = A \circ \epsilon = A$

$\emptyset \circ A = A \circ \emptyset = \emptyset$

More Useful Laws for Regular Expressions

$$(A \cup B) \circ C = (A \circ C) \cup (B \circ C)$$

$$A \circ (B \cup C) = (A \circ B) \cup (A \circ C)$$

$$(A^*)^* = A^*$$

$$\emptyset^* = \epsilon^* = \epsilon$$

$$(\epsilon \cup A)^* = A^*$$

$$(A \cup B)^* = (A^* B^*)^*$$

Limitations of Finite Automata

Cannot look ahead.

Fixed number of bits of memory.

How many bits to recognise this, without lookahead?

$$\{0^n1^n \mid n \geq 0\} = \{\epsilon, 01, 0011, 000111, \ldots\}$$

Exercise: Is the language $L_1 = \{0^n1^n \mid 0 \leq n \leq 999999999\}$ regular?

What about $L_2 = \left\{ w \;\middle|\; \begin{array}{l} w \text{ has an equal number of occurrences} \\ \text{of the substrings 01 and 10} \end{array} \right\}$?

The Pumping Lemma for Regular Languages

Lemma

If A is a regular language over Σ, then there is some integer p such that, for all $s \in A$ of length at least p, there exist $x, y, z \in \Sigma^*$ such that $s = xyz$ and

1. $xy^i z \in A$ *for all* $i \geq 0$, *and*
2. $|y| > 0$, *and*
3. $|xy| \leq p$.

This is the standard tool for proving languages non-regular.

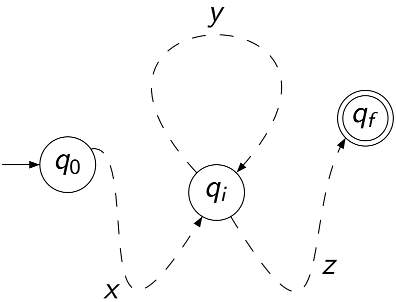
Loosely, it says that if we have a regular language A and consider a sufficiently long string $s \in A$, then a recogniser for A must traverse some loop to accept s. So A must contain infinitely many strings exhibiting repetition of some substring in s.

Intuition for the Pumping Lemma

Pigeonhole principle: If you put p pigeons into fewer than p holes, some hole has more than one pigeon.

If a DFA has p states, and you run it on a string longer than p symbols, it must enter some state twice.

Therefore it passes through a cycle in the graph!



Tools for the Proof

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA.

Definition

Let $\hat{\delta} : Q \times \Sigma^* \to Q$ such that for all $q \in Q$, $s \in \Sigma^*$ and $a \in \Sigma$,

$$\hat{\delta}(q, \epsilon) = q,$$
$$\hat{\delta}(q, as) = \hat{\delta}(\delta(q, a), s).$$

Lemma

M accepts a string s if and only if $\hat{\delta}(q_0, s) \in F$.

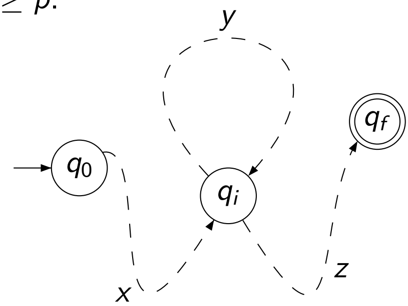
Lemma

For all $q \in Q$ and $x, y \in \Sigma^*$,

$$\hat{\delta}(q, xy) = \hat{\delta}(\hat{\delta}(q, x), y).$$

Proving the Pumping Lemma

Let DFA $M = (Q, \Sigma, \delta, q_0, F)$ recognise A.
Let $p = |Q|$ and consider s with $|s| \geq p$.
In an accepting run for s, some state must be re-visited. Let q_i be the first such state. At the first visit, x has been consumed, at the second, xy, (strictly longer than x). This suggests a way of splitting s into x, y and z such that $xz, xyz, xyyz, \ldots$ are all in A.



Notice that $y \neq \epsilon$. Also, if input consumed has length k then the number of state visits is $k + 1$. Let $m + 1$ be the number of state visits when reading xy, then $|xy| = m \leq p$. Notice that $m \leq p$, because $m + 1$ is the number of state visits with only one repetition.

Using the Pumping Lemma

The pumping lemma says:

$$A \text{ regular} \Rightarrow \exists p \forall s \in A \exists x, y, z \in \Sigma^* : \left\{ \begin{array}{l} s \text{ can be written} \\ xyz \text{ such that} \ldots \end{array} \right.$$

We can use its contrapositive to show that a language is non-regular:

$$\forall p \exists s \in A \forall x, y, z \in \Sigma^* : \left\{ \begin{array}{l} s \text{ can't be written} \\ xyz \text{ such that} \ldots \end{array} \right\} \Rightarrow A \text{ not regular}$$

Coming up with such an s is sometimes easy, sometimes difficult.

Pumping Example 1

We show that $B = \{0^n 1^n \mid n \geq 0\}$ is not regular.

Assume it is, and let p be the pumping length.

Consider $0^p 1^p \in B$ with length greater than p.

By the pumping lemma, $0^p 1^p = xyz$, with $xy^i z$ in B for all $i \geq 0$.

But y cannot consist of all 0s, since $xyyz$ then has more 0s than 1s.

Similarly y cannot consist of all 1s. And if y has at least one 0 and one 1, then some 1 comes before some 0 in $xyyz$.

So we inevitably arrive at a contradiction if we assume that B is regular.

Pumping Example 2

$C = \{w \mid w \text{ has an equal number of 0s and 1s}\}$ is not regular.

Assume it is, and let p be the pumping length.

Consider $0^p1^p \in C$ with length greater than p.

By the pumping lemma, $0^p1^p = xyz$, with xy^iz in C for all $i \geq 0$, $y \neq \epsilon$, and $|xy| \leq p$. Since $|xy| \leq p$, y consists entirely of 0s.

But then $xyyz \notin C$, a contradiction.

A simpler alternative proof: If C were regular then also B from before would be regular, since $B = C \cap 0^*1^*$ and regular languages are closed under intersection.

Pumping Example 3

Show that $D = \{ww \mid w \in \{0,1\}^*\}$ is not regular.

Assume it is, and let p be the pumping length.

Consider $0^p10^p1 \in D$ with length greater than p.

By the pumping lemma, $0^p10^p1 = xyz$, with xy^iz in D for all $i \geq 0$, $y \neq \epsilon$, and $|xy| \leq p$.

Since $|xy| \leq p$, y consists entirely of 0s.

But then $xyyz \notin D$, a contradiction.

Example 4 – Pumping Down

We show that $E = \{0^i 1^j \mid i > j\}$ is not regular.

Assume it is, and let p be the pumping length.

Consider $0^{p+1}1^p \in E$.

By the pumping lemma, $0^{p+1}1^p = xyz$, with xy^iz in E for all $i \geq 0$, $y \neq \epsilon$, and $|xy| \leq p$.

Since $|xy| \leq p$, y consists entirely of 0s.

But then $xz \notin E$, a contradiction.