Finite automata and formal languages (DIT322, TMV028)

Nachiappan V.,
based on slides by Thomas Sewell
and Nils Anders Danielsson

2020-02-10

Today

- Converting regular expressions to finite automata.
- More regular expression algebra.
- *Closure properties* of regular languages.
- Technique for proving that languages are not regular.

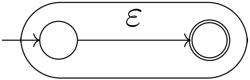
Converting REs to FA

Converting REs to FA

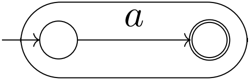
Given a regular expression e, we can construct an ε-NFA by structural recursion on e.

$\emptyset$

ε

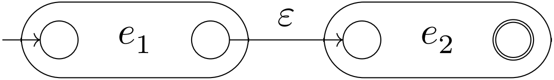


$a \in \Sigma$



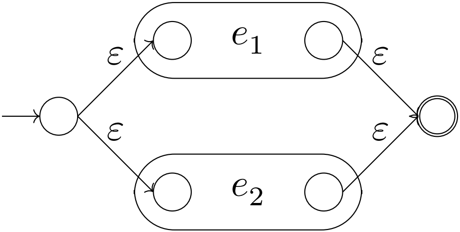
Converting REs to FA: e_1e_2

e_1e_2



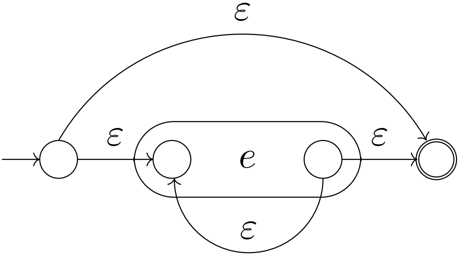
Converting REs to FA: $e_1 + e_2$

$e_1 + e_2$

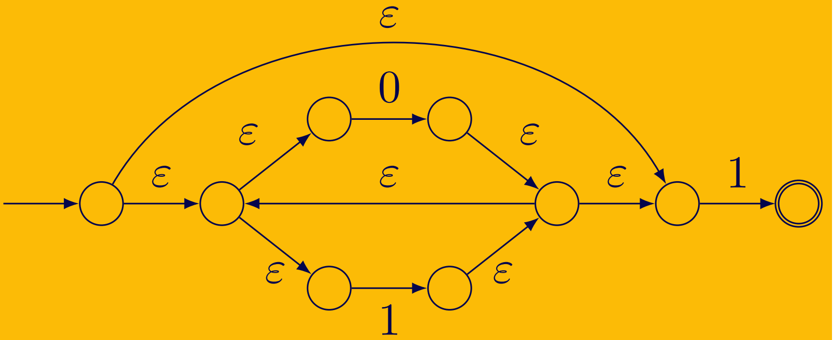


Converting REs to FA: e^*

e^*



Which RE converts to the following ε-NFA?



1. $(0 + 1)1$.
2. $01 + 1$.
3. $(0^* + 1^*)1$.
4. $(0 + 1)^*1$.

Regular Expression Algebra

Regular Expression Algebra

Recall from earlier:

- $e_1 = e_2$ if $L(e_1) = L(e_2)$.
- Algebraic laws for $\emptyset$, ε, a, $e_1 + e_2$, and $e_1 e_2$.

What about e^*?

Laws of the Closure Operator *

- $(e^*)^* = e^*$
- $\emptyset^* = \varepsilon$
- $\varepsilon^* = \varepsilon$
- $ee^* = e^*e$
- $e_1(e_2e_1)^* = (e_1e_2)^*e_1$ (called *Shifting*)
- $(e_1^*e_2)^*e_1^* = (e_1 + e_2)^*$ (called *Denesting*)

Which of the following equalities hold? You may consider the alphabet $\{a, b\}$ if needed.

1. $e^*e^* = e^*$.
2. $(e_1 + e_2)^* = e_1^* + e_2^*$.
3. $e^* = ee^* + \varepsilon$.
4. $(\varepsilon + \emptyset)^* = \varepsilon$.

Disproving RE Equalities, *quickly*!

How do we disprove $(e_1 + e_2)^* = e_1^* + e_2^*$?

- Replace expression variables with letters from the alphabet: e_1 with a, and e_2 with b.
- Refute the equality $(a + b)^* = a^* + b^*$:
  - $ab \in L((a + b)^*)$ but $ab \notin L(a^* + b^*)$,
  - hence $L((a + b)^*) \neq L(a^* + b^*)$,
  - hence $(a + b)^* \neq a^* + b^*$.
- Rejoice in cleverness of constructing a counter-example ☺.

Closure Properties

Closure Properties of Regular Languages

Given two regular languages L_1 and L_2,

- $L_1 \cup L_2$ is regular
- $L_1 \cap L_2$ is regular
- $\overline{L_1}$ and $\overline{L_2}$ are regular

i.e., regular languages are *closed* under these operations.

Proving Closure Properties

Proof for closure of regular languages under $\cap$:

- Given two regular languages L_1 and L_2, and hence their respective DFAs A_1 and A_2, construct the product DFA $A_1 \otimes A_2$.
- $L(A_1 \otimes A_2)$
  $= L(A_1) \cap L(A_2)$
  $= L_1 \cap L_2$
- $L(A_1 \otimes A_2)$ is regular, hence so is $L_1 \cap L_2$. $\square$

Proving Closure Properties

Similarly, to show that regular languages are closed under $\cup$ and $\overline{}$, we use the corresponding DFA constructions $\oplus$ and $\overline{}$.

Given that L_1, L_2, and L_3 are regular, which of the following languages are also regular?

1. $L_1 \cup (L_2 \cap L_3)$
2. $L_1 - L_2$
3. $\overline{\overline{L_1}}$
4. ${L_1}^*$

Proving Closure Properties using REs

Some closure properties can also be proved using regular expressions:

- Given that L is regular, it must have a corresponding regular expression e.
- e^* is a valid regular expression, and by its semantics, L^* is also regular.

The Pumping Lemma

Proving Languages are *not* Regular

- Some languages, such as $\{0^n 1^n | n \geq 1\}$, are not regular.
- Intuitively, this is because FAs have a finite number of states and cannot remember an arbitrary number of input symbols.
- But how do we show this?

Proving Languages are *not* Regular

Let's prove that $L = \{0^n 1^n | n \geq 1\}$ is not regular.

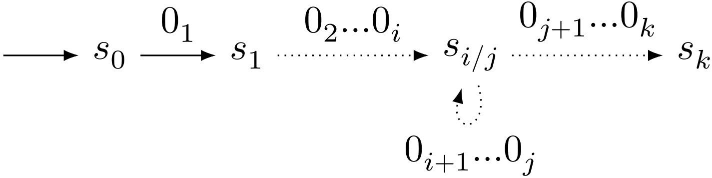
- Suppose that L is regular. Then there must exist a DFA A with some k states s.t. $L(A) = L$.
- $0^k 1^k \in L$, hence there must exist a sequence of transitions:

$$\longrightarrow s_0 \xrightarrow{0_1} s_1 \xrightarrow{0_2} s_2 \xrightarrow{0_3 \ldots 0_k} s_k$$

- Notice that the sequence involves $k+1$ state variables.

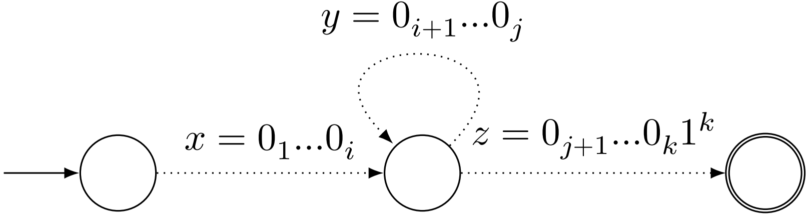
Proving Languages are *not* Regular

Since A only has k states, by the pigeon hole principle, some state must be "visited twice": $s_i = s_j$ for some distinct i and j.



Proving Languages are *not* Regular

Thus the DFA A must be of the form:



Notice that the word xyz is accepted as expected, but so are the words xz, $xyyz$, $xyyyz$,..., etc.

Proving Languages are *not* Regular

- The words xz, $xyyz$, $xyyyz$..., etc., are accepted by A, but are not in L since they don't have the same number of 0s and 1s.
- Contradicts the fact that $L(A) = L$, hence our assumption must be wrong.
- Therefore, L is not regular. $\square$

The Pumping Lemma

- The Pumping Lemma provides a convenient generalization of the previous proof as a property that all regular languages must have.
- We can use it as a tool to argue by contradiction that a given language is not regular.

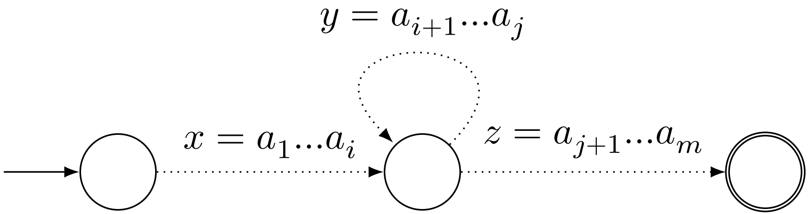
The Pumping Lemma, informally

"Informally, it says that all sufficiently long words in a regular language may be *pumped*—that is, have a middle section of the word repeated an arbitrary number of times—to produce a new word that also lies within the same language." - Wikipedia

The Pumping Lemma, precisely

Given L is regular, there exists a constant n such that for all words w of length m with $m \geq n$, we have $w = xyz$ such that:

- $|y| > 0$
- $|xy| = j$ s.t. $j \leq n$
- $\forall k \geq 0.\ xy^k z \in L$



Which of the following languages are *not* regular? The alphabet is $\{0, 1\}$. If you suspect that a language is not regular, use the pumping lemma to verify by contradiction.

1. Words with equal number of 0s and 1s.
2. $\{0^n 1 0^n | n \geq 1\}$.

Today

- Regular expressions to finite automata.
- RE laws involving the closure operator.
- Closure properties of regular languages.
- Pumping lemma for regular languages.