CS:4330 Theory of Computation

Spring 2018

Regular Languages

Equivalences between NFAs, DFAs, and REs

Haniel Barbosa

The University of Iowa

Readings for this lecture

Chapter 1 of [Sipser 1996], 3rd edition. Sections 1.2 and 1.3.

Equivalence of NFA and DFA

Theorem

Every NFA has an equivalent DFA to recognize the same language

Proof idea: Build an equivalent DFA

Let $N = (Q, \Sigma, \delta, q_0, F)$ be the NFA recognizing the language A. We construct the DFA M recognizing A

- For any $R \subseteq Q$, the ϵ-closure of R is

$$E(R) = R \cup \{q \in Q \mid q \text{ reachable from } R \text{ through } 0 \text{ or more } \epsilon\text{-transitions}\}$$

- M is such that
  1. $Q' = \mathscr{P}(Q)$
  2. $\delta'(R, \mathrm{a}) = \bigcup_{r \in R, R \subseteq Q} E(\delta(r, \mathrm{a}))$
  3. $q_0' = E(\{q_0\})$
  4. $F' = \{R \subseteq Q \mid R \text{ contains a final state of } N\}$

Equivalence of NFA and DFA

Still necessary to prove that N and M have the same languages:

N accepts a word w if and only if M accepts w

Equivalence of NFA and DFA

Still necessary to prove that N and M have the same languages:

$$N \text{ accepts a word } w \text{ if and only if } M \text{ accepts } w$$

Proof idea: structural induction on the length of w

- ▷ Base case: $|w| = 0$
- ▷ Induction step: Assume it holds for $|w| = n$, prove for $|w| = n + 1$

Equivalence of NFA and DFA

Still necessary to prove that N and M have the same languages:

$$N \text{ accepts a word } w \text{ if and only if } M \text{ accepts } w$$

Proof idea: structural induction on the length of w

- ▷ Base case: $|w| = 0$
- ▷ Induction step: Assume it holds for $|w| = n$, prove for $|w| = n+1$

Hint: use the following definitions, for any $w \in \Sigma^*$

- ▷ For an NFA N:
  $\hat{\delta}(q, w)$ as the set of states reached by executing N on w starting from q
  - ▶ resulting set is in $\mathscr{P}(Q)$
- ▷ For the corresponding DFA M:
  $\hat{\delta}(q', w)$ as the state reached by executing M on w starting from q'
  - ▶ resulting state is in $Q' = \mathscr{P}(Q)$

Language recognized by an automaton

Corollary

A language is regular if and only if some NFA recognizes it.

- ▷ If a language A is recognized by an NFA then A is recognized by the equivalent DFA, therefore A is regular

- ▷ If a language A is regular, it means that it is recognized by a DFA. But any DFA is also an NFA, therefore the language is recognized by an NFA

Application of NFA: proving Regular Languages are closed under...

Given two regular languages A and B

- ▷ Union: $A \cup B = \{w \mid w \in A \text{ or } w \in B\}$
- ▷ Intersection: $A \cap B = \{w \mid w \in A \text{ and } w \in B\}$
- ▷ Concatenation: $A \circ B = \{vw \mid v \in A \text{ and } w \in B\}$
- ▷ Complementation: $\bar{A} = \{w \mid w \notin A\}$
- ▷ Reverse: $A^R = \{a_1 a_2 \dots a_k \mid a_k a_{k-1} \dots a_1 \in A\}$
- ▷ Star: $A^* = \{w_1 w_2 \dots w_k \mid k \geq 0 \text{ and each } w_i \in A\}$

Closure under Reverse

Reverse: $A^R = \{a_1 a_2 \dots a_k \mid a_k a_{k-1} \dots a_1 \in A\}$

Theorem

For any regular language L, L^R is also regular

Proof idea

Given M that recognizes A, what if you could "run it backwards"?

- ▷ Build M^R as M but with all arrows reversed and accept state interchanged with start state
- ▷ Note that M^R *is an NFA*

Application of NFA: proving equivalence with regular expressions

Theorem

A language is regular if and only if some regular expression describes it.

Proof ideas

1. If a language A is described by a regular expression R then A is recognized by an NFA, therefore A is regular

   There is an NFA N such that N recognizes $\mathscr{L}(R)$

2. If a language A is regular, it means that it is recognized by a DFA. Then we can always deduce a regular expression from it.

   Turn DFA into equivalent regular expression

Part 1: From regular expressions to NFAs

By induction on the length of R:

- ▷ Base cases (R has length 1):
  - $R = \{\mathrm{a}\}$
  - $R = \epsilon$
  - $R = \varnothing$

Part 1: From regular expressions to NFAs

By induction on the length of R:

- Base cases (R has length 1):
  - $R = \{\mathrm{a}\}$
  - $R = \epsilon$
  - $R = \varnothing$

- Inductive case: let R have length $k > 1$. Assume that for any smaller regular expression, there is an NFA.
  R may be one of the following cases:
  - $R = R_1 \cup R_2$
  - $R = R_1 R_2$
  - $R = (R_1)^*$

Part 2: From DFAs to regular expressions

1. Define Generalized Nondeterministic Finite Automaton (GNFA in short).
   Instead of $\delta : Q \times \Sigma \to Q$, we use $\delta : Q \times RE \to Q$

   Arrows labelled with regular expressions, with one start and one accept state

2. Show how to convert any DFA to an equivalent GNFA

3. Show an algorithm to convert any GNFA to an equivalent GNFA with 2 states

4. Convert a 2-state GNFA to an equivalent RE.