EECS 2001A : Introduction to the Theory of Computation

Suprakash Datta

Course page: `http://www.eecs.yorku.ca/course/2001`
Also on Moodle

The Big Question

Note that

- NFA can solve every problem that DFA can (DFA are also NFA)
- Can DFA solve every problem that NFA can?
- In other words: Are NFA more powerful than DFA?

The Surprising Answer

We will prove that

- every NFA is equivalent to a DFA (with upto exponentially more states)
- Non-determinism does not help FA's to recognize more languages!
- NFAs recognize regular languages
- Corollary: DFAs and NFAs can be used interchangeably to solve problems or study properties of regular languages

Terminology: ϵ-closure

- Let $N = (Q, \Sigma, \delta, q_0, F)$ be any NFA
- Consider any set $R \subseteq Q$
- Define $E(R) = \{q | q \text{ can be reached from a state in } R \text{ by following 0 or more } \epsilon\text{-transitions}\}$
- $E(R)$ is the ϵ- closure of R under ϵ-transitions

Equivalence of DFA, NFA

- Statement: For all languages $L \subseteq \Sigma^*$, $L = L(N)$ for some NFA N <u>if and only if</u> $L = L(M)$ for some DFA M
- One direction is easy: A DFA M is also a NFA N. So N does not have to be "constructed" from M
- The other direction: Construct M from N

Equivalence of DFA, NFA - A Special Case

Given $N = (Q, \Sigma, \delta, q_0, F)$, construct $M = (Q', \Sigma, \delta', q'_0, F')$ so that for any $w \in \Sigma^*$, M accepts w if and only if N accepts w.
First a special case: Assume that NFA N has no ϵ-transitions

- Need to keep track of each subset of Q
- So $Q' = \mathcal{P}(Q), q'_0 = \{q_0\}$
- $\delta'(R, a) = \cup(\delta(r, a))$ over all $r \in R, R \in Q'$
- $F' = \{R \in Q' | R \text{ contains an accept state of } F\}$

Next: let us assume that ϵ-transitions are used in N

Equivalence of DFA, NFA - The General Case

- $Q' = \mathcal{P}(Q)$
- $q_0' = E(\{q_0\})$
- for all $R \in Q'$ and $a \in \Sigma$
  $\delta'(R, a) = \{q \in Q | q \in E(\delta(r, a)) \text{ for some } r \in R\}$
- $F' = \{R \in Q' | R \text{ contains an accept state of } N\}$

Why This Construction Works...

for any string $w \in \Sigma^*$,

- can argue informally that w is accepted by N iff w is accepted by M
- Can prove using induction on the number of steps of computation

Closure: Revisiting Old Terminology

A set is defined to be closed under an operation if that operation on members of the set always produces a member of the same set. E.g.:

- The integers are closed under addition, multiplication
- The integers are not closed under division
- Σ^* is closed under concatenation
- A set can be defined by closure – Σ^* is called the (Kleene) closure of Σ under concatenation.

New Terminology: Regular Operations

The regular operations are:

- Union
- Concatenation
- Star (Kleene Closure): For a language A, define $A^* = \{w_1 w_2 w_3 \dots w_k | k \geq 0, \text{ and each } w_i \in A\}$

Want to prove that regular languages are closed under regular operations

Proving Closure under Regular Operations

We showed that regular languages are closed under:

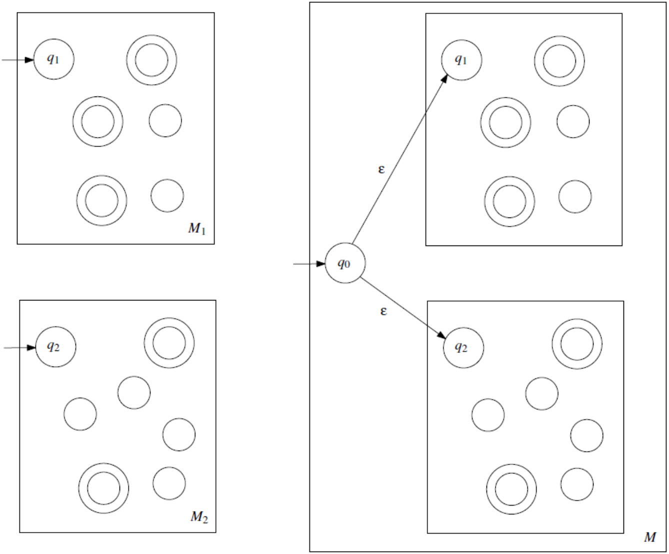
- Complementation
- Union

We got stuck at concatenation, and introduced nondeterminism

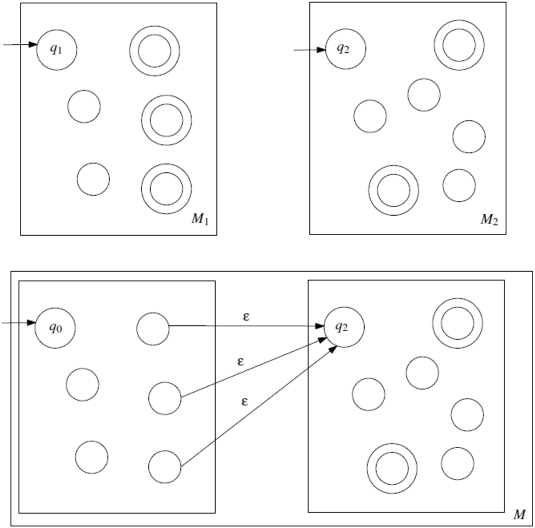
Next, we show closure under

- Union (easier proof)
- Concatenation
- Star (Kleene Closure)

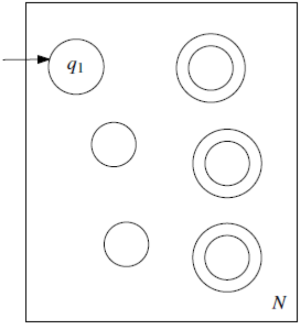
Proving Closure Under Union

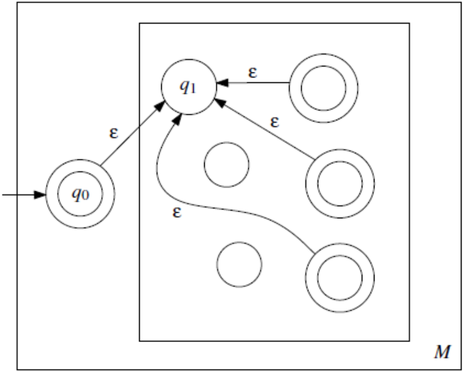


Proving Closure Under Concatenation



Proving Closure Under Kleene Star





Incorrect reasoning about RL

- Since $L_1 = \{w|w = a^n, n \in \mathbb{N}\}$, $L_2 = \{w|w = b^n, n \in \mathbb{N}\}$ are regular, therefore $L_1 \cdot L_2 = \{w|w = a^n b^n, n \in \mathbb{N}\}$ is regular

- If L_1 is a regular language, then $L_2 = \{w^R|w \in L_1\}$ is regular, and therefore $L_1 \cdot L_2 = \{ww^R|w \in L_1\}$ is regular

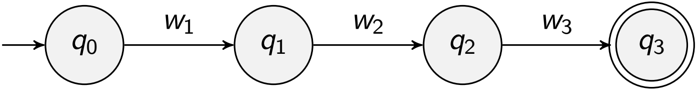
Putting it all together

A recursive definition for regular languages

- $\emptyset$, $\{\epsilon\}$ and $\{a\}$ for any symbol $a \in \Sigma$ are regular languages
- If L_1 and L_2 are regular languages, then $L_1 \cup L_2$, $L_1 L_2$ and L_1^* are regular languages.
- Nothing is a regular language unless it is obtained from the above two clauses.

Every Finite Language is Recognized by a NFA

- Given a word $w = w_1 w_2 \ldots w_k$ there is a NFA that recognizes $\{w\}$. Example of $w = w_1 w_2 w_3$



- Use the union construction on languages containing single words...

Regular Languages: Exercises

- Prove the following result:
  If L_1 and L_2 are regular languages, then $L_1 \cap \overline{L_2}$ is a regular language too
- Describe the language that is recognized by this NFA: