Lec 06. Properties of Regular Languages

Eunjung Kim

Questions to examine

1. Given an NFA M, decide if $L(M) = \emptyset$ or not.
2. Given two regular languages L_1 and L_2, decide if $L_1 = L_2$.
3. Is $\mathit{Prefix}(L)$ is regular when L is regular?
4. How about $\mathit{Suffix}(L)$?
5. Quotient of L by a symbol $a \in \Sigma$, denoted by L/a, is regular when L is?
6. How about $a \setminus L$?
7. Fix a DFA M and a state $s \in Q$. The set of all strings w such that the (accepting) computation history of w visits the state s, is it regular?
8. Fix a DFA M. The set of all strings w such that the (accepting) computation history of w visits all the state of M, is it regular?

Deciding if $L = \emptyset$

Given a regular language L, we want to decide if $L = \emptyset$ or not.

L is given by NFA N

$L(M) \neq \emptyset$ if and only if there is a directed path from the initial state q_0 to OOOOOOOOOO in the transition diagram of N.

Recall: $w \in \Sigma^*$ satisfies $\delta^*(q_0, w) = q$ if and only if there is a (q_0, q)-walk in the transition diagram labelled by w (ϵ-label allowed).

Deciding if $L = \emptyset$

Given a regular language L, we want to decide if $L = \emptyset$ or not.
You can convert R into an NFA and apply the previous criteria, or do the following.

L is given by a regular expression R

If there is no occurrence of $\emptyset$ in R, $L(R) \neq \emptyset$.

Otherwise, check if $L(R) = \emptyset$ inductively:

1. $L(R_1 \cup R_2) = \emptyset$ if and only if $L(R_1) = \emptyset$ and $L(R_2) = \emptyset$.
2. $L(R_1 \cdot R_2) = \emptyset$ if and only if $L(R_1) = \emptyset$ or $L(R_2) = \emptyset$.
3. $L(R^\star) \neq \emptyset$ (even when $R = \emptyset$).

When L is regular, so is $Prefix(L)$?

Given two strings $x, w \in \Sigma^*$, x is a prefix of w if $w = xy$ for some $y \in \Sigma^*$.
For a language $L \subseteq \Sigma^*$, let $Prefix(L) = \{x \in \Sigma^* : x \text{ is a prefix of } w \in L\}$.

If L is regular, $Prefix(L)$ is regular

- $w \in L$ can be written as $w = xy$ if and only if $\delta^*(q_0, x) = q$ for some state $q \in Q$ such that ....
- Let $L_q = \{x \in \Sigma^* : \delta^*(q_0, x) = q\}$ for each $q \in Q$. Is L_q regular?
- $Prefix(L) = \bigcup_{q \in Q \text{ such that....}} L_q$.

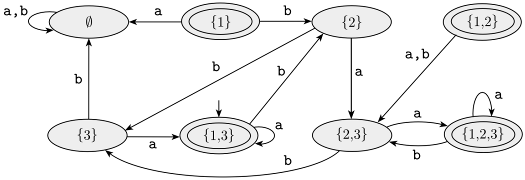

Figure 1.43, Sipser 2012

When L is regular, so is $Suffix(L)$?

Given two strings $x, w \in \Sigma^*$, x is a suffix of w if $w = yx$ for some $y \in \Sigma^*$. For a language $L \subseteq \Sigma^*$, let $Suffix(L) = \{x \in \Sigma^* : x \text{ is a suffix of } w \in L\}$.

If L is regular, $Suffix(L)$ is regular

- $w \in L$ can be written as $w = yx$ if and only if $\delta^*(q, x) \in F$ for some state $q \in Q$ such that ....
- Let $A_q = \{x \in \Sigma^* : \delta^*(q, x) \in F\}$. Is A_q regular?
- $Suffix(L) = \bigcup_{q \in Q \text{ such that....}} A_q$

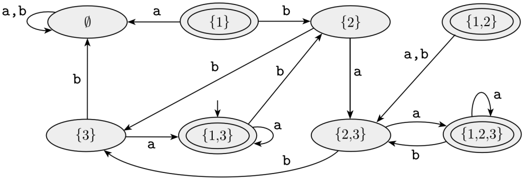

Figure 1.43, Sipser 2012

Reverse language is regular

Let $A_q = \{x \in \Sigma^* : \delta^*(q, x) \in F\}$. Is A_q regular?

Yes, A_q is the same language as $L_q(\overleftarrow{M})$,

- where $\overleftarrow{M}$ is the reversal of M,
- modified to have a unique initial state by adding ϵ-transitions to the accept states of M.
- $\overleftarrow{M}$ recognizes precisely the language

$$rev(L(M)) := \{\text{reverse of w} : w \in L(M)\}$$

When L is regular, so is $\mathit{Suffix}(L)$?

Given two strings $x, w \in \Sigma^*$, x is a suffix of w if $w = yx$ for some $y \in \Sigma^*$.
For a language $L \subseteq \Sigma^*$, let $\mathit{Suffix}(L) = \{x \in \Sigma^* : x \text{ is a suffix of } w \in L\}$.

If L is regular, $\mathit{Suffix}(L)$ is regular

More succinctly,

- $\mathit{Suffix}(L) = \mathit{rev}(\mathit{Prefix}(\mathit{rev}(L)))$.
- As the class of regular languages is closed under reverse and Prefix operations, it is closed closed under Suffix operation.

QUOTIENT L/a FOR $a \in \Sigma$

Given a language L over Σ and a symbol $a \in \Sigma$, the quotient of L by a denoted as L/a is the language

$$\{x \in \Sigma^* : xa \in L\}.$$

Is L/a regular?

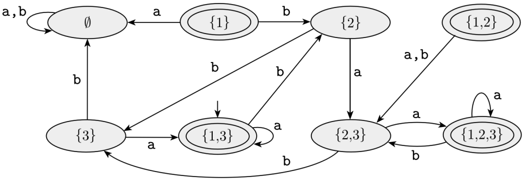

Figure 1.43, Sipser 2012

- For a state $q \in Q$, if $x \in L_q$ satisfies $xa \in L$, then for all $y \in L_q$ we have $ya \in L$.
- That is, $L_q \subseteq L/a$ or $L_q \cap L/a = \emptyset$.
- How to tell if $L_q \in L/a$?

The language $a \setminus L$ for $a \in \Sigma$

Given a language L over Σ and a symbol $a \in \Sigma$, the language $a \setminus L$ is defined as

$$\{x \in \Sigma^* : ax \in L\}.$$

Is $a \setminus L$ regular?

Idea: Express $a \setminus L$ using the operations we examined so far to immediately conclude.

More exotic language P_s

- Fix a DFA M and a state $s \in Q$.
- Let P_s be the set of all string $w \in L$ such that the accepting computation history of w visits the state s.
- Is P_s regular?

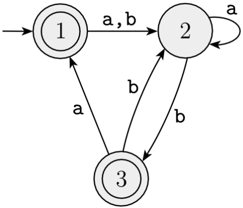

Figure 1.21, Sipser 2012

More exotic language P_s

First approach.

- For any string w, $w \in P_s$ if and only if it can be written as $w = xy$ with $\delta^*(q_0, x) = s$ and $\delta^*(s, y) \in F$.
- That is $P_s = L_s \cdot A_s$, where L_s and A_s are.....(we've seen previously).

More exotic language P_s

Second approach: use Myhill-Nerode Theorem.

Myhill-Nerode Theorem

L is regular if and only if the number of equivalence classes of $\equiv_L$ is finite.

Idea: use the DFA M recognizing L to identify the equivalence relation $\equiv_{P_s}$, (or a refinement of it) of finite index.

- For $Z \subseteq Q$ and $q \in W$, let $L_{Z,q}$ be the set of all strings w such that the computation history of w on M visits precisely the states in Z and end in q.
- $\Sigma^* = \dot{\bigcup}_{Z \subseteq Q, q \in Z} L_{W,q}$.
- We want to argue that any strings $x, y \in L_{Z,q}$ are indistinguishable by P_s. But for proving this claim, we need to try *all strings z which might potentially distinguish x and y... or do we?*

More exotic language P_s

Second approach: use Myhill-Nerode Theorem and test for a finite number of extensions z (and argue that it suffices).

Myhill-Nerode Theorem, in action

P_s is regular if for any $Z \subseteq Q$ and $q \in Z$,

- any $x, y \in L_{Z,q}$ are indistinguishable by P_s, or equivalently
- for any $x, y \in L_{Z,q}$ and for any $z \in \Sigma^*$, $xz \in P_s$ if and only if $yz \in P_s$.

What are the key property of z which will make $xz \in P_s$ (or not) for $x \in L_{Z,q}$?

More exotic language P_s

Second approach: use Myhill-Nerode Theorem and test for a finite number of extensions z (and argue that it suffices).

Myhill-Nerode Theorem, in action

P_s is regular if for any $Z \subseteq Q$ and $q \in Z$,

- any $x, y \in L_{Z,q}$ are indistinguishable by P_s, or equivalently
- for any $x, y \in L_{Z,q}$ and for any $z \in \Sigma^*$, $xz \in P_s$ if and only if $yz \in P_s$.

What are the key property of z which will make $xz \in P_s$ (or not) for $x \in L_{Z,q}$?

1. whether $\delta^*(q, z) \in F$ or not: this dictates whether $xz \in L$.
2. whether the states visited by the computation history of $\delta^*(q, z)$ include s or not: this affects whether the computation history of xz from q_0 visits s or not.

A bit more exotic language

Fix a DFA M. The set of all strings w such that the (accepting) computation history of w visits all the state of M, is it regular?

Even More exotic language

Why do we care about the second approach using Myhill-Nerode theorem when the first approach seems much simpler?

Even more exotic language. Fix two states s_1, s_2 of a DFA M. Let P_{s_1,s_2} be the set of strings $w \in L$ whose computation history visits both s_1, s_2 and visiting s_2 only after visiting s_1.

Is P_{s_1,s_2} regular?

What we learned so far

- Finite (state) automata: a machine with limited memory.
- Nondeterministic FA has the extra feature of making multiple transitions in parallel and ϵ-transition. Conversions between DFA and NFA possible (no added power).
- Regular expression: describes the 'shape' of a regular language directly.
- Conversion between regular expression and NFA using Generalized NFA.
- The class of regular languages is closed under various operations such as: union, concatenation, kleene star, intersection, complement, suffix/prefix, reverse, quotient, etc...
- Pumping lemma as a tool to prove that a language is nonregular.
- Myhill-Nerode Theorem as a powerful characterization of regular languages.
- One can prove various properties of NFA/DFA and regular language combining the tools we learned.