CSE547T Class 6

Jeremy Buhler

February 6, 2017

1 What Have We Wrought?

At this point, we have some nifty little machines, the ε-NFAs (equivalent to NFAs, equivalent to DFAs).

- The point of this whole exercise is to decide how powerful the finite automaton model of computing is.
- So far, we have not characterized the set of languages that can be accepted by a finite automaton.
- Is it all possible languages? (No – we proved that)
- Can we find a decent characterization of which languages are accepted?
- "Decent": reasonably intuitive, and w/o reference to the machines themselves.

2 Regular Languages

Remember the constructive method we sketched out for building languages? Let's define that a little more formally.

Defn: a *regular language* over the alphabet Σ is defined recursively as follows.

1. $\emptyset$ is a regular language.
2. $\{\varepsilon\}$ is a regular language.
3. For all $a \in \Sigma$, $\{a\}$ is a regular language.
4. If L_1 and L_2 are regular languages, so is $L_1 \cdot L_2$.
5. If L_1 and L_2 are regular languages, so is $L_1 \cup L_2$.
6. If L is a regular language, so is L^*.

We normally describe regular languages using *regular expression* notation, which is basically the above except with parentheses to disambiguate complex constructions.

- Let R be the function mapping regular languages to regular expressions.
- Here is the definition of R:

Language	Expression
$\emptyset$	$\emptyset$
$\{\varepsilon\}$	ε
$\{a\}$	a
$L_1 \cdot L_2$	$(R(L_1) \cdot R(L_2))$ or $(R(L_1)R(L_2))$
$L_1 \cup L_2$	$(R(L_1) \cup R(L_2))$
L^*	$(R(L)^*)$

- Usual operator precedence is *, then $\cdot$, then $\cup$.
- We normally drop the parentheses where operator precedence resolves any ambiguity.
- Example: $10^* \cup 11^*1$ means $((1(0^*)) \cup ((1(1^*))1))$.

Some quick examples:

- $L = \{x \in \{0,1\}^* \mid x \text{ ends with } 110\}$.
- Can describe L by expression $(0 \cup 1)^*110$. Proof is trivial.
- $L = \{x \in \{0,1\}^* \mid x \text{ has an even number of 1s}\}$.
- Can describe L by expression $(0^*10^*1)^*0^*$. Proof is slightly less trivial.
- $L = \{x \in \{0,1\}^* \mid |x| \text{ is even}\}$.
- One way to describe it: $((0 \cup 1)(0 \cup 1))^*$.
- Another way: $(00 \cup 01 \cup 10 \cup 11)^*$.
- Notice that there is not in general a *unique* regular expression for a regular language.

NB: if you are used to defining regular expressions with `grep`, our regexps must match the *entire* string, not just a substring of it!

3 Kleene's Theorem, Part I

Regular languages turn out to be a good way to describe what a finite automaton can do.

- **Theorem** (Kleene): the set of languages that can be accepted by a finite automaton is exactly the set of regular languages!
- Need to prove two directions.
- Part one: given a regular language L, can build a finite automaton M such that $L(M) = L$.
- Note that it does not matter whether we use a DFA, an NFA, or an ε-NFA – all three kinds of machines are equivalent.
- We will build an ε-NFA.

- To show that we can handle *all* regular languages, we proceed by structural induction on their definition.
- **Base cases**: need machines accepting $\emptyset$, $\{\varepsilon\}$, and $\{a\}$. Here they are:

- **Inductive 1**: Let $M_1 = (Q_1, \Sigma, q_0^1, A_1, \delta_1)$ and $M_2 = (Q_2, \Sigma, q_0^2, A_2, \delta_2)$ be machines for L_1 and L_2. Build a machine M_{12} for $L_1 \cdot L_2$.
- $Q = Q_1 \cup Q_2$.
- $q_0 = q_0^1$; $A = A_2$. (start at first machine, end only at second)
- To construct δ, start with all transitions defined by δ_1 together with δ_2, then add null transitions from every state of A_1 to q_0^2.
- Picture:

- **Inductive 2**: Let $M_1 = (Q_1, \Sigma, q_0^1, A_1, \delta_1)$ and $M_2 = (Q_2, \Sigma, q_0^2, A_2, \delta_2)$ be machines for L_1 and L_2. Build a machine M_{12} for $L_1 \cup L_2$.
- $Q = Q_1 \cup Q_2 \cup \{q_u\}$. Last state is new.
- $q_0 = q_u$, $A = A_1 \cup A_2$
- To construct δ, start with all transitions defined by δ_1 together with δ_2, then add null transitions from q_u to q_0^1 and q_0^2.
- Picture:

- **Inductive 3**: Let $M_1 = (Q_1, \Sigma, q_0^1, A_1, \delta_1)$ be a machine for L_1. Build a machine M_{12} for L_1^*.

- $Q = Q_1 \cup \{q_k\}$. Last state is new.
- $q_0 = q_k$, $A = \{q_k\}$.
- To construct δ, start with δ_1. Add a null transition from q_k to q_0^1. Finally, add null transitions from states of A_1 back to q_k.
- Picture:

- *interesting note*: this construction uses fact that $\emptyset^* = \{\varepsilon\}$.
- Conclude that any regular language L has a mapping to an equivalent ε-NFA.

4 Kleene's Theorem, Part II

OK, we're halfway there. There is a finite automaton for every regular language. Is there a regular language for every finite automaton?

- Part two: given a finite automaton M, there exists a regular language L equivalent to $L(M)$.
- This time, we'll start from a DFA.
- Will build up L inductively using a sneaky trick!
- Suppose DFA M has n states. Number states arbitrarily from 1 to n.
- **Defn**: $L(p, q, j)$ is the set of strings that label paths in M (of any length) that start at state p, end at state q, and whose intermediate states (i.e. not the endpoints) have numbers at most j.
- Quick picture:

- **Claim**: $L(p, q, j)$ is regular for all choices of p, q, and $j \geq 0$.
- Proof by induction on j.
- **Bas**: if $j = 0$, the path $p \to q$ cannot have any intermediate states, since all states of M are numbered ≥ 1. Hence, if the path exists, it must have no edges ($p = q$) or one edge.

    1. If $p = q$, $L(p, q, 0)$ includes ε.

2. If there exists an edge from p to q labeled with character $a \in \Sigma$, $L(p, q, 0)$ includes a.
3. No other strings can be included in $L(p, q, 0)$.

- Hence, $L(p, q, 0)$ is a language with at most $|\Sigma| + 1$ strings. All finite languages are regular (build by direct union of strings).

Now for the fun inductive bit:

- **Ind**: Want to show that $L(p, q, j + 1)$ is regular.
- Every path in M from p to q that passes through states numbered at most $j+1$ has the following form:

- Can divide path into one or more segments. First segment goes from p to $j + 1$. Intermediate segments go from $j + 1$ back to $j + 1$ (that is, they are cycles). Final segment goes from $j + 1$ to q.
- Since all states on path are numbered at most $j + 1$, all intermediate states of each segment are numbered at most j.
- (Note that path may never touch $j + 1$ at all – one segment).
- I claim that

$$L(p, q, j + 1) = L(p, q, j) \cup L(p, j + 1, j)L(j + 1, j + 1, j)^* L(j + 1, q, j).$$

- For each string $x \in L(p, q, j + 1)$, we can segment it according to which part of the path generates which part of x.
- If only one segment (we never touch $j + 1$), then $x \in L(p, q, j)$.
- Otherwise, The parts of x match the components of the three-part concatenation above.
- Conversely, every string in the RHS language follows a path that goes from p to q and passes through states numbered at most $j + 1$, so it is also in $L(p, q, j + 1)$.
- Finally, the sublanguages used to define $L(p, q, j + 1)$ are regular by IH, and they are combined using only union, concatenation, and Kleene closure. Hence, their combination is also regular!

OK, almost done.

- Let q_0 be the start state of M, and let A be its accepting state set.
- Let $L = \bigcup_{r \in A} L(q_0, r, n)$.

- L is a finite union of regular languages, hence regular.
- Moreover, $x \in L$ iff it labels a path from start state of M to an accepting state (path may pass through *any* intermediate states), which is true iff $x \in L(M)$.
- Conclude that $L = L(M)$. QED

5 Closure Properties of RLs

Regular languages are closed under (finite) union, concatenation, and Kleene closure by definition. What about other natural set operations?

Lemma: The set of regular languages is closed under complement.

- If L is a regular language, I claim that $\overline{L} = \Sigma^* - L$ is regular.
- Let M be a DFA for L. Construct a new machine $\overline{M}$ by reversing the accepting and nonaccepting states of M.
- Can prove that $\overline{M}$ accepts $\Sigma^* - L$.

Lemma: The set of regular languages is closed under intersection.

- If L_1, L_2 are regular languages, I claim that $L_1 \cap L_2$ is regular.
- let $M_1 = (Q_1, \Sigma, q_0^1, A_1, \delta_1)$ and $M_2 = (Q_2, \Sigma, q_0^2, A_2, \delta_2)$ be DFAs for L_1 and L_2.
- Construct new machine $M_{12} = (Q_{12}, \Sigma, q_0^{12}, A_{12}, \delta_{12})$ as follows (cross-product construction).
- $Q_{12} = \{\langle q_1, q_2 \rangle \mid q_1 \in Q_1, q_2 \in Q_2\}$
- $q_0^{12} = \langle q_0^1, q_0^2 \rangle$
- $A_{12} = \{\langle q_1, q_2 \rangle \mid q_1 \in A_1, q_2 \in A_2\}$
- $\delta_{12}(\langle q_1, q_2 \rangle, a) = \langle \delta_1(q_1, a), \delta_2(q_2, a) \rangle$
- Can prove that M_{12} accepts $L_1 \cap L_2$.