The Pumping Lemma for RLs

Theorem 4.1
(The pumping lemma for regular languages.)
Let $L$ be a regular language. Then there exists a constant $n$ (which depends on $L$) such that for every string $w \in L$ such that $|w| \geq n$, $w = xyz$, such that:

1. $y \neq \epsilon$
2. $|xy| \leq n$
3. for all $k \geq 0$, $xy^kz \in L$

Proof:

Since $|Q| = n$, by the pigeonhole principle, one cannot find $n+1$ different $p_i$'s for $0 \leq i \leq n$ to be distinct.

Find $i \neq j$, where $0 \leq i < j \leq n$, such that $p_i = p_j$.

Note: $p_0 = q_0$
The Pumping Lemma for RLs

**Theorem 4.1 (Pumping lemma, continued)**

**PROOF:**

Break \( w = xyz \) as

\[
\begin{align*}
  x &= a_1 a_2 w a_i \\
  y &= a_i a_{i+1} a_{i+2} w a_j \\
  z &= a_{j+1} a_{j+2} w a_m
\end{align*}
\]

From Figure 4.1 of IATLC, Hopcroft, Motwani, & Ullman, 2001.

**Case 1:** \( k = 0 \).

Then, \( w = xy^kz = a_1 a_2 w a_i a_{i+1} a_{j+2} w a_m \). \( A \) goes from \( p_0 = q_0 \) to \( p_i \) on prefix \( x \).

Since \( p_i = p_j \), then \( A \) continues from \( p_i \) to \( p_m \in F \) on suffix \( z \). Thus, \( A \) accepts \( w = xz \).

**Case 2:** \( k > 0 \).

Then, \( A \) goes from \( p_0 = q_0 \) to \( p_i \) on \( x \), circles from \( p_i \) to \( p_j \) \( k \) times on \( y^k \), and then to \( p_m \in F \) on \( z \).

Thus, for any \( k \geq 0 \), \( xy^kz \) is also accepted by \( A \); that is, \( xy^kz \in L \).
The Pumping Lemma for RLs

Use of PL, continued...

4. Applying the PL, we know \( w \) can be broken into \( xyz \), satisfying the PL properties.
   - again, we may not know how to break \( w \), so we use \( x, y, z \) as parameters.

5. Derive a contradiction by picking \( i \) such that \( xy^iz \notin L \).
   - \( i \) might depend on parameter \( n, x, y, \) and/or \( z \).

Example, continued ...

\( L = \{ w \in \{0 \}^* \text{, where } |w| \text{ is a square} \} \)

Claim: \( L \) not regular.

Suppose \( L \) regular. Then \( L = L(A) \) for some DFA \( A = (Q, \Sigma, q_0, F) \); so, let \( n = |Q| \).

Consider \( w = 0^{n^2} \in L \). Then \( w = xyz \), where \( |xy| \leq n \) and \( y \neq \epsilon \).

By PL, \( xyyz \in L \); but, \( n^2 < |xyyz| \leq n^2 + n \).

Since we have derived a contradiction, the only unproved assumption — that \( L \) is regular — must be at fault. Therefore, \( L \) is not regular.

Example:

\( L = \{ w \in \{0 \}^* \text{, where } |w| \text{ is a square} \} \)

Claim: \( L \) not regular.

Suppose \( L \) regular. Then \( L = L(A) \) for some DFA \( A = (Q, \Sigma, q_0, F) \); so, let \( n = |Q| \).

Consider \( w = 0^{n^2} \in L \). Then \( w = xyz \), where \( |xy| \leq n \) and \( y \neq \epsilon \).

By PL, \( xyyz \in L \); but, \( n^2 < |xyyz| \leq n^2 + n \).

Thus \( |xyyz| \) is not square, so \( xyyz \notin L \).

Since we have derived a contradiction, the only unproved assumption — that \( L \) is regular — must be at fault. Therefore, \( L \) is not regular.
The Pumping Lemma for RLs

The Pumping Lemma can be good for you ...

Closure Properties of RLs

Boolean Operations

Theorem 4.4 (Closure under Union)
If $L$ and $M$ are RLs, then so is $LM$.

Theorem 4.5 (Closure under Kleene star)
If $L$ is RL, then so is $L^*$.

Theorem 4.8 (Closure under Intersection)
If $L$ and $M$ are RLs, then so is $LM$.

Theorem 4.10 (Closure under Difference)
If $L$ and $M$ are RLs, then so is $L \setminus M$.  

Theorem 4.7 (Closure under Complementation)
If $L$ is RL over $\Sigma$, then so is $L = \Sigma^* \setminus L$.
Chapter 4: Properties of Regular Languages

Closure Properties of RLs

- Reversal
  - Definition
    - The reversal of a string \( w = a_1 a_2 \cdots a_n \), denoted \( w^R \), is the string \( w \) written backwards as \( a_n a_{n-1} \cdots a_1 \).
  - Theorem 4.11 (Closure under Reversal)
    - If \( L \) is RL, then so is \( L^R \).

From Figure 4.2 of IATLC, Hopcroft, Motwani, & Ullman, 2001. 

Automation for \( \{0,1\}^* - L(A) \), where \( L(A) = (0+1)^* 01 \).

Closure Properties of RLs

- Substitution
  - Take a regular language \( L \) over some alphabet \( \Sigma \).
  - For each \( a \in \Sigma \), let \( L_a \) be a regular language.
  - Let \( s \) be the substitution defined by \( s(a) = L_a \) for each \( a \).
    - Extend \( s \) to strings by \( s(a_1 a_2 \cdots a_n) = s(a_1) s(a_2) \cdots s(a_n) \); i.e., concatenate the languages \( L_{a_1} L_{a_2} \cdots L_{a_n} \).
    - Extend \( s \) to languages by \( s(M) = \bigcup_{w \in M} s(w) \).
    - Then \( s(L) \) is regular.
Closure Properties of RLs

**Homomorphisms**

- A string homomorphism is a function on strings that works by substituting a particular string for each symbol.

**Theorem 4.14**

- If \( L \) is a regular language over alphabet \( \Sigma \), and \( h \) is a homomorphism on \( \Sigma \), the \( h(L) \) is also regular.

- Extending \( s \) to strings by \( s: \Sigma^* \times \Sigma^* \).

- Extend \( s \) to languages by \( s: \Sigma^* \times \Sigma^* \).

Note: \( R_i \)'s are regular expressions.
Closure Properties of RLs

Theorem 4.16

If \( h \) is a homomorphism from alphabet \( \Sigma \) to alphabet \( \Gamma \), and \( L \) is a regular language over \( \Gamma \), then \( h^{-1}(L) \) is also a regular language.

Constructive Proof of Theorem 4.16

Given a DFA \( A \) for RL \( L \).

Recall \( h: \Sigma \times \Gamma \to \Gamma \)

\( A = (Q, \Sigma, \delta, s_A, F) \)

So, \( B = (Q, \Sigma, \gamma, s_B, F) \)

where \( \gamma(q, a) = \Delta(q, h(a)) \)

Decision Properties of RLs

Converting among Representations

- NFA-to-DFA, \( O(n^3 2^n) \)
  - Compute \( \varepsilon \)-closure in \( O(n^3) \); for each of the \( 2^n \) states in DFA, compute transitions by consulting the \( \varepsilon \)-closure and NFA’s transition table in \( O(n^3) \).
- DFA-to-NFA, \( O(n) \)
  - Modify transition table to be on sets (and \( \varepsilon \))

Converting among Representations, continued ...

- FA-to-RE, \( O(n^3 4^n) \)
  - Generate \( n^2 \) expressions \( n \) times where size of the RE constructed can quadruple in each round.
- RE-to-FA, \( O(n) \)
  - RE of length \( n \); parse RE into expression tree and then use \( \varepsilon \)-NFA construction algorithm.
Decision Properties of RLs

Testing Emptiness of RLs
- Choose DFA representation.
- Use a graph reachability algorithm to test if at least one accepting state is reachable from the start state.
- Note that reachability calculations take no more than $O(n^2)$ if the automaton has $n$ states.

Testing Membership in a RL
- Choose DFA representation.
- Simulate the DFA on input $w$.

Testing an RL’s Finiteness
- Every finite language is regular (why?).
- A regular language is not necessarily finite.
- DFA $A$ with cycles $\Rightarrow L(A)$ is infinite.
- RE $E$, presence of $*$ almost always means infinite, except for annihilators and $\varepsilon^*$.

Equivalence & Minimization

Testing Equivalence of States
- Real goal is testing equivalence of representations of two regular languages.
- Interesting fact: DFAs have unique (up to state names) minimum-state equivalents.
- States $p, q \in Q$ of DFA $A$ are equivalent if
  - For all $w$, $\Delta(p, w)$ is an accepting state if and only if $\Delta(q, w)$ is an accepting state.
  - Note that $\Delta(p, w)$ and $\Delta(q, w)$ do not have to be the same state.
  - If two states are not equivalent, then they are distinguishable.

Example:

From Figure 4.8 of JTL, Hopcroft, Motwani, & Ullman, 2001.
Equivalence & Minimization

Testing Equivalence of States, continued ...

- **Table-filling algorithm** (via recursive discovery)
  - BASIS: \( p \in F, q \notin F \Rightarrow \{ p, q \} \) distinguishable
  - INDUCTION: Let \( p, q \in Q \) such that for some \( a \in \Sigma \), \( r = \delta(p, a) \) and \( s = \delta(q, a) \) are distinguishable. Then, \( \{ p, q \} \) distinguishable.
  - \( \exists w \in \Sigma^* \) that distinguishes \( r \) from \( s \); i.e., either \( \Delta(r, w) \in F \) or \( \Delta(s, w) \in F \), but not both. Then, string \( aw \) must distinguish \( p \) from \( q \) since \( \Delta(p, aw) = \Delta(r, w) \) and \( \Delta(q, aw) = \Delta(s, w) \).

---

Equivalence & Minimization

- **Theorem 4.20**
  - If two states are not distinguished by the table-filling algorithm, then the states are equivalent.
  - **Testing Equivalence of RLs**
    - Given regular languages \( L_1 \) and \( L_2 \)
    - Convert each representation to a DFA
    - Consider DFA \( A \) where \( L(A) = L_1 \cap L_2 \)
    - Use the table-filling algorithm to test if \( \{ s_1, s_2 \} \) are equivalent; if so, \( L_1 = L_2 \)

---

Equivalence & Minimization

- **Example**: (See Figure 4.8 or Slide 32)
  - \( x \) indicates pairs of distinguishable states and a blank square indicates equivalence
  - \( \checkmark \) indicates final states

---

Equivalence & Minimization

- **Example**:
Equivalence & Minimization

Minimization of DFAs

- For each DFA there is an equivalent DFA that has as few states as any DFA accepting the same language. Further, this minimum-state DFA is unique for the language.
- State equivalence partitions the set of states.

Example:

From Figure 4.11 of *IATLC*, Hopcroft, Motwani, & Ullman, 2001.
Partitions: 
- \{A, E\}
- \{B, H\}
- \{C\}
- \{D, F\}
- \{G\}

Theorem 4.23

The equivalence of states is transitive.

Theorem 4.24

If we create for each state \(q\) of a DFA a block consisting of \(q\) and all the states equivalent to \(q\), then the different blocks of states form a partition of the set of states.
Equivalence & Minimization

**DFA minimization algorithm** \( A = (Q_A, \Sigma, \delta_A, s_A, F_A) \)

1. Use the *table-filling algorithm* to find all pairs of equivalent states.
2. Partition the set of states \( Q_A \) into blocks of mutually exclusive states by the method described above.
3. Construct the minimum-state equivalent DFA \( B \) by using the blocks as its states.
   - \( s_B \) is the block containing \( s_A \).
   - \( F_B \) is the set of blocks containing \( f \in F_A \).

**Theorem 4.26**

If \( A \) is a DFA and \( M \) the DFA constructed from \( A \) by the *DFA minimization algorithm*, then \( M \) has as few states as any DFA equivalent to \( A \).