Extended RE's

UNIX pioneered the use of additional operators and notation for RE's:

- \( E? = \) 0 or 1 occurrences of \( E = \epsilon + E \).
- \( E^+ = \) 1 or more occurrences of \( E = EE^* \).
- **Character classes** \( [a-zGX] = \) the union of all (ASCII) characters from \( a \) to \( z \), plus the characters \( G \) and \( X \), for example.

Algebraic Laws for RE's

If two expressions \( E \) and \( F \) have no variables, then \( E = F \) means that \( L(E) = L(F) \) (not that \( E \) and \( F \) are identical expressions).

- **Example:** \( 1^+ = 11^* \).

If \( E \) and \( F \) are RE's with variables, then \( E = F \) (\( E \) is equivalent to \( F \)) means that whatever languages we substitute for the variables (provided we substitute the same language everywhere the same variable appears), the resulting expressions denote the same language.

- **Example:** \( R^+ = RR^* \).

With two notable exceptions, we can think of union (+) as if it were addition with \( \emptyset \) in place of the identity 0, and concatenation, with \( \epsilon \) in place of the identity 1, as multiplication.

- + and concatenation are both associative.
- + is commutative.
- Laws of the identities hold for both.
- \( \emptyset \) is the annihilator for concatenation.

The exceptions:

1. Concatenation is **not** commutative: \( ab \neq ba \).
2. + is **idempotent**: \( E + E = E \) for any expression \( E \).

Checking a Law

Suppose we are told that the law \((R + S)^* = (R^* S^*)^* \) holds for RE's. How would we check that this claim is true?

- Think of \( R \) and \( S \) as if they were single symbols, rather than placeholders for languages, i.e., \( R = \{0\} \) and \( S = \{1\} \).
  - Then the left side is clearly “any sequence of 0’s and 1’s.”
• The right side also denotes any string of 0’s and 1’s, since 0 and 1 are each in \( L(0^*1^*) \).

• That test is necessary (i.e., if the test fails, then the law does not hold).

• We have particular languages that serve as a counterexample.

• But is it sufficient (if the test succeeds, the law holds)?

Proof of Sufficiency

The book has a fairly simple argument for why, when the “concretized” expressions denote the same language, then the languages we get by substituting any languages for the variables are also the same.

• But if you think that’s obvious, the book also has an example of “RE’s with intersection” where the same statement is false.

• Also — is it clear that we can tell whether two RE’s without variables denote the same language?

• Algorithm to do so will be covered.

Closure Properties

• Not every language is a regular language.

• However, there are some rules that say “if these languages are regular, so is this one derived from them.”

• There is also a powerful technique — the pumping lemma — that helps us prove a language not to be regular.

• Key tool: Since we know RE’s, DFA’s, NFA’s, \( \epsilon \)-NFA’s all define exactly the regular languages, we can use whichever representation suits us when proving something about a regular language.

Pumping Lemma

If \( L \) is a regular language, then there exists a constant \( n \) such that every string \( w \) in \( L \), of length \( n \) or more, can we written as \( w = xyz \), where:

1. \( 0 < |y| \).
2. \( |xy| \leq n \).
3. For all $i \geq 0$, $w y^i z$ is also in $L$.
   ✦ Note $y^i = y$ repeated $i$ times; $y^0 = \varepsilon$.
   ✦ The alternating quantifiers in the logical statement of the PL makes it very complex: $(\forall L)(\exists n)(\forall w)(\exists x, y, z)(\forall i)$.

Proof of Pumping Lemma

- Since we claim $L$ is regular, there must be a DFA $A$ such that $L = L(A)$.
- Let $A$ have $n$ states; choose this $n$ for the pumping lemma.
- Let $w$ be a string of length $\geq n$ in $L$, say $w = a_1a_2\cdots a_m$, where $m \geq n$.
- Let $q_i$ be the state $A$ is in after reading the first $i$ symbols of $w$.
  ✦ $q_0 =$ start state, $q_1 = \delta(q_0, a_1)$, $q_2 = \delta(q_1, a_1a_2)$, etc.
- Since there are only $n$ different states, two of $q_0, q_1, \ldots, q_n$ must be the same; say $q_i = q_j$, where $0 \leq i < j \leq n$.
- Let $x = a_1\cdots a_i$; $y = a_i+1\cdots a_j$; $z = a_{j+1}\cdots a_m$.
- Then by repeating the loop from $q_i$ to $q_i$ with label $a_{i+1}\cdots a_j$ zero times once, or more, we can show that $xy^iz$ is accepted by $A$.

PL Use

We use the PL to show a language $L$ is not regular.

- Start by assuming $L$ is regular.
- Then there must be some $n$ that serves as the PL constant.
  ✦ We may not know what $n$ is, but we can work the rest of the “game” with $n$ as a parameter.
- We choose some $w$ that is known to be in $L$.
  ✦ Typically, $w$ depends on $n$.
- 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.
- We derive a contradiction by picking $i$ (which might depend on $n, x, y, \text{and/or } z$) such that $xy^iz$ is not in $L$. 

3
Example
Consider the set of strings of 0's whose length is a perfect square; formally \( L = \{ 0^i \mid i \) is a square\).
- We claim \( L \) is not regular.
- Suppose \( L \) is regular. Then there is a constant \( n \) satisfying the PL conditions.
- Consider \( w = 0^{n^2} \), which is surely in \( L \).
- Then \( w = xyz \), where \( |xy| \leq n \) and \( y \neq \varepsilon \).
- By PL, \( xyyz \) is in \( L \). But the length of \( xyyz \) is greater than \( n^2 \) and no greater than \( n^2 + n \).
- However, the next perfect square after \( n^2 \) is \((n + 1)^2 = n^2 + 2n + 1\).
- Thus, \( xyyz \) is not of square length and is not in \( L \).
- Since we have derived a contradiction, the only unproved assumption — that \( L \) is regular — must be at fault, and we have a “proof by contradiction” that \( L \) is not regular.

Closure Properties
Certain operations on regular languages are guaranteed to produce regular languages.
- Example: the union of regular languages is regular; start with RE's, and apply + to get an RE for the union.

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 \( s(a_1a_2\cdots a_n) = s(a_1)s(a_2)\cdots s(a_n) \); i.e., concatenate the languages \( L_{a_1}, L_{a_2}, \ldots, L_{a_n} \).
  - Extend \( s \) to languages by \( s(M) = \bigcup_{w \in M} s(w) \).
- Then \( s(L) \) is regular.

Proof That Substitution of Regular Languages Into a Regular Language is Regular
- Let \( R \) be a regular expression for language \( L \).
• Let $R_a$ be a regular expression for language $s(a) = L_a$, for all symbols $a$ in $\Sigma$.

• Construct a RE $E$ for $s(L)$ by starting with $R$ and replacing each symbol $a$ by the RE $L_a$.

• Proof that $L(E) = s(L)$ is an induction on the height of (the expression tree for) RE $R$.

Basis: $R$ is a single symbol, $a$. Then $E = R_a$, $L = \{a\}$, and $s(L) = s(\{a\}) = L(R_a)$.

• Cases where $R$ is $\epsilon$ or $\emptyset$ easy.

Induction: There are three cases, depending on whether $R = R_1 + R_2$, $R = R_1 R_2$, or $R = R_1^*$. We’ll do only $R = R_1 R_2$.

• $L = L_1 L_2$, where $L_1 = L(R_1)$ and $L_2 = L(R_2)$.

• Let $E_1$ be $R_1$, with each $a$ replaced by $R_a$, and $E_2$ similarly.

• By the IH, $L(E_1) = s(L_1)$ and $L(E_2) = s(L_2)$.

• Thus, $L(E) = s(L_1)s(L_2) = s(L)$.

Applications of the Substitution Theorem

• If $L_1$ and $L_2$ are regular, so is $L_1 L_2$.
  ♦ Let $s(a) = L_1$ and $s(b) = L_2$. Substitute into the regular language $\{ab\}$.

• So is $L_1 \cup L_2$.
  ♦ Substitute into $\{a, b\}$.

• Ditto $L_1^*$.
  ♦ Substitute into $L(a^*)$.

• Closure under homomorphism = substitution of one string for each symbol.
  ♦ Special case of a substitution.

Example: Homomorphism

Let $L = L(0^*1^*)$, and let $h$ be a homomorphism defined by $h(0) = aa$ and $h(1) = \epsilon$.

• Then $h(L) = L(aa^*) = \{\text{all strings of an even number of } a\text{'s}\}$.

Closure Under Inverse Homomorphism

• $h^{-1}(L) = \{w \mid h(w) \text{ is in } L\}$. 

5
See argument in course reader. Briefly:

✦ Given homomorphism \( h \) and regular language \( L \), start with a DFA \( A \) for \( L \).

✦ Construct DFA \( B \) for \( h^{-1}(L) \), by having \( B \) go from state \( q \) to state \( p \) on input \( a \) if \( \hat{\delta}(q, h(a)) = p \).

**Closure Under Reversal**

✦ The *reverse* of a string \( w = a_1a_2\cdots a_n \) is \( a_n\cdots a_2a_1 \).

✦ Denoted \( w^R \).

✦ Note \( \epsilon^R = \epsilon \).

✦ The reverse of a language \( L \) is the set containing the reverse of each string in \( L \).

✦ If \( L \) is regular, so is \( L^R \).

✦ Proof: use RE’s, recursive reversal as in course reader.