Closure Properties of Regular Languages

Union, Intersection, Difference, Concatenation, Kleene Closure, Reversal, Homomorphism, Inverse Homomorphism

Slides from Jeff Ullman

Closure Properties

Recall a closure property is a statement that a certain operation on languages, when applied to languages in a class (e.g., the regular languages), produces a result that is also in that class.

For regular languages, we can use any of its representations to prove a closure property.

Closure Under Union

If L and M are regular languages, so is \( L \cup M \).

**Proof:** Let L and M be the languages of regular expressions R and S, respectively.

Then \( R+S \) is a regular expression whose language is \( L \cup M \).

Closure Under Concatenation and Kleene Closure

Same idea:

- \( RS \) is a regular expression whose language is \( LM \).
- \( R^* \) is a regular expression whose language is \( L^* \).

Closure Under Intersection

If L and M are regular languages, then so is \( L \cap M \).

**Proof:** Let A and B be DFA's whose languages are L and M, respectively.

Construct C, the product automaton of A and B.

Make the final states of C be the pairs consisting of final states of both A and B.

Example: Product DFA for Intersection
Closure Under Difference

- If $L$ and $M$ are regular languages, then so is $L - M = \text{strings in } L \text{ but not } M$.
- **Proof:** Let $A$ and $B$ be DFA's whose languages are $L$ and $M$, respectively.
- Construct $C$, the product automaton of $A$ and $B$.
- Make the final states of $C$ be the pairs where $A$-state is final but $B$-state is not.

**Example: Product DFA for Difference**

![Diagram of DFA product]

Notice: difference is the empty language

Closure Under Complementation

- The complement of a language $L$ (with respect to an alphabet $\Sigma$ such that $\Sigma^*$ contains $L$) is $\Sigma^* - L$.
- Since $\Sigma^*$ is surely regular, the complement of a regular language is always regular.

Closure Under Reversal

- Recall example of a DFA that accepted the binary strings that, as integers were divisible by 23.
- We said that the language of binary strings whose reversal was divisible by 23 was also regular, but the DFA construction was very tricky.
- Good application of reversal-closure.

Closure Under Reversal – (2)

- Given language $L$, $L^R$ is the set of strings whose reversal is in $L$.
- **Example:** $L = \{0, 01, 100\}$; $L^R = \{0, 10, 001\}$.
- **Proof:** Let $E$ be a regular expression for $L$.
- We show how to reverse $E$, to provide a regular expression $E^R$ for $L^R$.

Reversal of a Regular Expression

- **Basis:** If $E$ is a symbol $a$, $\epsilon$, or $\emptyset$, then $E^R = E$.
- **Induction:** If $E$ is
  - $F + G$, then $E^R = F^R + G^R$.
  - $FG$, then $E^R = G^RF^R$.
  - $F^*$, then $E^R = (F^R)^*$.
Example: Reversal of a RE

- Let $E = 01^* + 10^*$. 
  - $E^R = (01^* + 10^*)^R = (01^*)^R + (10^*)^R$
  - $= (1^*)^R0 + (0^*)^R1^R$
  - $= (1^*)^R0 + (0^*)^R1$
  - $= 1^*0 + 0^*1$.

Homomorphisms

- A **homomorphism** on an alphabet is a function that gives a string for each symbol in that alphabet.
  - **Example**: $h(0) = ab$; $h(1) = \epsilon$.
  - Extend to strings by $h(a_1...a_n) = h(a_1)...h(a_n)$. **Note**: $h(\epsilon) = \epsilon$.
  - **Example**: $h(01010) = ababab$.

Closure Under Homomorphism

- If $L$ is a regular language, and $h$ is a homomorphism on its alphabet, then $h(L) = \{h(w) \mid w \text{ is in } L\}$ is also a regular language.
  - **Proof**: Let $E$ be a regular expression for $L$.
    - Apply $h$ to each symbol in $E$.
    - Language of resulting RE is $h(L)$.

Example: Closure under Homomorphism

- Let $h(0) = ab$; $h(1) = \epsilon$.
  - Let $L$ be the language of regular expression $01^* + 10^*$.
  - Then $h(L)$ is the language of regular expression $ab\epsilon^* + \epsilon(ab)^*.$
  - **Note**: use parentheses to enforce the proper grouping.

Example – Continued

- $ab\epsilon^* + \epsilon(ab)^*$ can be simplified.
  - $\epsilon^* = \epsilon$, so $ab\epsilon^* = ab\epsilon$.
  - $\epsilon$ is the identity under concatenation.
    - That is, $\epsilon E = E \epsilon = E$ for any RE $E$.
  - Thus, $ab\epsilon^* + \epsilon(ab)^* = ab\epsilon + \epsilon(ab)^* = ab + (ab)^*$.
  - Finally, $L(ab)$ is contained in $L((ab)^*)$, so a RE for $h(L)$ is $(ab)^*$.

Inverse Homomorphisms

- Let $h$ be a homomorphism and $L$ a language whose alphabet is the output language of $h$.
  - $h^{-1}(L) = \{w \mid h(w) \text{ is in } L\}$. 
Example: Inverse Homomorphism

- Let \( h(0) = ab \); \( h(1) = \varepsilon \).
- Let \( L = \{ab, baba\} \).
- \( h^{-1}(L) \) is the language with two 0's and any number of 1's = \( L(1^*01^*01^*) \).

Notice: no string maps to baba; any string with exactly two 0's maps to abab.

Proof – (2)

- The transitions for \( B \) are computed by applying \( h \) to an input symbol \( a \) and seeing where \( A \) would go on sequence of input symbols \( h(a) \).
- Formally, \( \delta_B(q, a) = \delta_A(q, h(a)) \).

Example: Inverse Homomorphism

Construction

- Start with a DFA \( A \) for \( L \) (mapped-to symbols).
- Construct a DFA \( B \) for \( h^{-1}(L) \) with:
  - The same set of states.
  - The same start state.
  - The same final states.
  - Input alphabet = the symbols to which homomorphism \( h \) applies (original input symbols)

Proof – (3)

- By induction on \(|w|\)
- \( IH(n) \): for all \( w \) of length \( n \),
  \( \delta_B(q_0, w) = \delta_A(q_0, h(w)) \).
- Basis: Show \( IH(0): \ w = \varepsilon \)
- \( \delta_B(q_0, \varepsilon) = q_0 \) and
  \( \delta_A(q_0, h(\varepsilon)) = \delta_A(q_0, \varepsilon) = q_0 \).

Inductive step: assume \( n \geq 0 \) and \( IH(n) \) to show \( IH(n+1) \).

- Let \( w = xa \) be any string of length \( n+1 \).
- \( \delta_B(q_0, w) = \delta_B(\delta_B(q_0, x), a) \).
  = \( \delta_B(\delta_A(q_0, h(x)), a) \) by \( IH(n) \).
  = \( \delta_A(\delta_A(q_0, h(x)), h(a)) \) by def. of DFA \( B \).
  = \( \delta_A(q_0, h(x)h(a)) \) by def. of extended \( \delta \).
  = \( \delta_A(q_0, h(w)) \) by def. of homomorphism.
Summary: Models for Regular Languages

- 4 "models": DFA, NFA, $\varepsilon$-NFA, RE
- Transition diagrams/transition tables
- Automatic conversions between them
  - Subset construction (with greedy version)
  - Removing $\varepsilon$-transitions
  - Building $\varepsilon$-NFA from RE
  - Building RE from DFA (splicing paths, elim. States)
- Algebra of Regular Expressions

Summary: Properties

- Pumping Lemma to show non-regularity
- Basic closure properties: Union, concatenation, Kleene-*, complement, reversal, intersection, difference
- Closure under homomorphism & inverse
- Tests for: membership, emptiness, infiniteness, equivalence, containment
- How to find a minimum state DFA

Blue uses cross product construction