

Lecture 6: Finishing Regular expressions

David Dill

Department of Computer Science

Outline

- More closure properties
- Minimization of DFAs.
- Decision problems on regular languages.

String Homomorphisms

Start with function $h: \Sigma_1 \rightarrow \Sigma_2^*$. Then extend it to strings:

Base: $h(\epsilon) = \epsilon$

Induction: $h(xa) = h(x)h(a)$.

So, if $h(0) = ab$ and $h(1) = \epsilon$ then $h(0110) = abab$

h can also be applied to entire languages: $h(L) = \{h(x) \mid x \in L\}$.

Closure under Homomorphisms

Theorem: If $L \subseteq \Sigma_1^*$ is regular and $h: \Sigma_1 \rightarrow \Sigma_2^*$ is a string homomorphism, then $h(L)$ is regular.

Construction: There is a regular expression R for L . Substitute $h(a)$ for each occurrence of each a in R .

The proof is by induction on the structure of regular expressions.

In the induction cases, you have to prove things like

$$h(L^*) = [h(L)]^*$$

Inverse Homomorphism

$$h^{-1}(L) = \{x \mid h(x) \in L\}.$$

Consider h that deletes “ c ”: $h(a) = a, h(b) = b, h(c) = \epsilon$.

Consider $L = (abc)^*$. What is $h(L)$?

What is $h^{-1}[(ab)^*]$? Answer: $c^*(c^*ac^*bc^*)^*$ – *not the same language*.

$h^{-1}(L)$ is the *largest* language L' such that $h(L') = L$.

Closure under Inverse Homomorphism

Theorem: If $L \subseteq \Sigma_2^*$ is regular and h is a homomorphism from $\Sigma_1 \rightarrow \Sigma_2^*$, then $h^{-1}(L)$ is regular.

Construction:

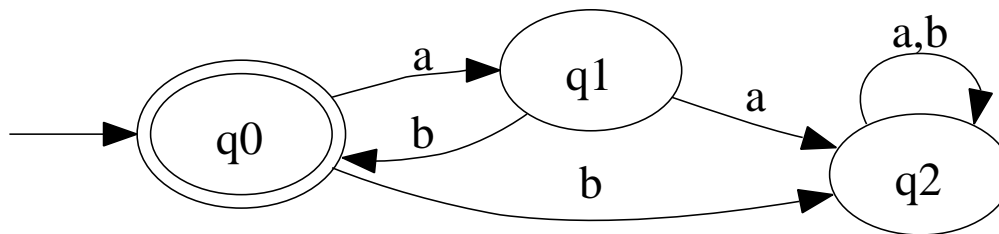
Let $(Q, \Sigma_2, \delta, q_0, F)$ be a DFA accepting L . We construct $(Q, \Sigma_1, \gamma, q_0, F)$ such that

$$\gamma(q_i, a) = \hat{\delta}(q_i, h(a))$$

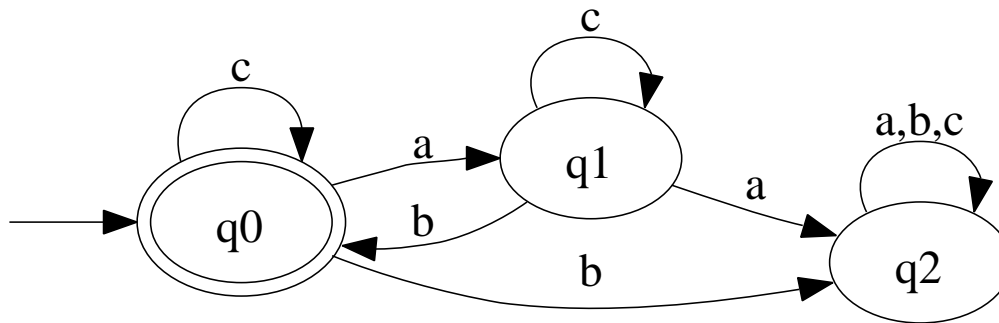
Rest of proof: Show $\hat{\gamma}(q_i, x) = \hat{\delta}(q_i, h(x))$ for all $x \in \Sigma_1^*$.

Example application of Inverse Homomorphism Construction

Try it on “delete c ” with $(ab)^*$. Here is a DFA for $(ab)^*$



Here is the result of the construction for the inverse homomorphism:



This is the same language as the regular expression of a few slides ago.

Closure Properties for Non-Regularity

Proof by contradiction: If we can construct a non-regular language using regularity-preserving operations, one of the original languages must have been non-regular.

Example: The language L over $\Sigma = \{0, 1, 2\}$ having equal numbers of 0's and 1's is not regular.

proof: Suppose the language were regular. Define h such that $h(0) = 0$, $h(1) = 1$, and $h(2) = \epsilon$. Then $h(L)$ would be regular. But $h(L)$ is the language over $\{0, 1\}$ having equal numbers of 0's and 1's, proved non-regular above, so L must not be regular.

Example: The language of strings over $\Sigma = \{0, 1\}$ having different numbers of 0's and 1's is not regular.

proof By contradiction: Suppose the language were regular. Then its complement would be, too, since regular languages are closed under complementation.

But the complement of this language is the language of strings having the same number of 0's and 1s, which was proved non-regular earlier in the lecture using the pumping lemma.

So the language in question must not be regular.

Minimum DFA

One of the coolest things about regular languages is that every language has a *unique* minimum-state DFA that accepts it.

Theorem $L_1 = L_2$ iff the minimum state DFA accepting L_1 is the same as the minimum DFA accepting L_2 .

This gives a nice decision procedure for equivalence of regular expressions or NFAs:

1. Convert each to a DFA.
2. Minimize the DFAs.
3. See if they are the same.

Minimization

Two states in one (or even two DFAs) are *equivalent* if the same languages are accepted from those states.

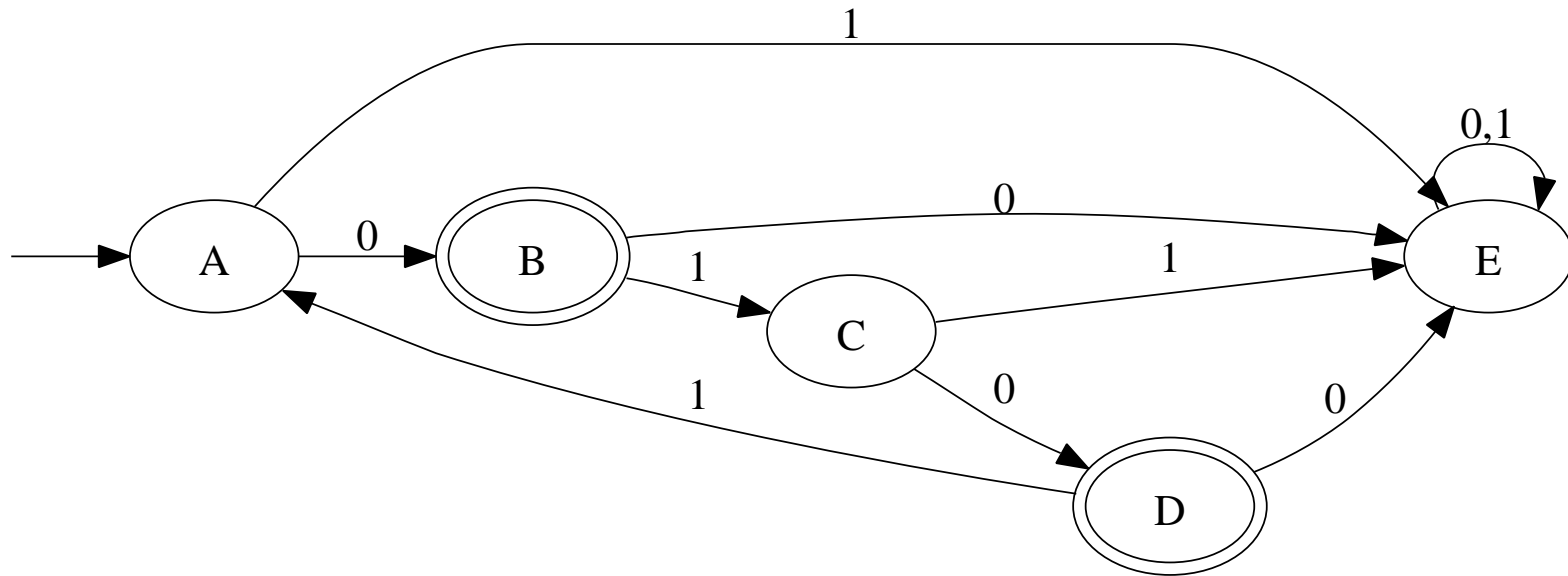
$$\forall x \in \Sigma^* (\hat{\delta}(q_1, x) \in F \Leftrightarrow \hat{\delta}(q_2, x) \in F)$$

If two states are *not equivalent*, there must be a string x that is accepted from one state and not the other.

x is a *distinguishing input* for q_1 and q_2 .

Example DFA

In the following, states A and B are *not equivalent*, and 0 is a distinguishing input. B and D are *equivalent*.



Minimizing a DFA

Problem: Given a DFA, find a minimum DFA.

Wrong approach: Try to find pairs of equivalent states.

Right approach: Start assuming everything is equivalent. Make them inequivalent only when forced to.

1. Mark states p and q as “not equivalent” only if one is final and the other is not. (ϵ distinguishes them.)
2. Whenever states p and q are not yet marked as “not equivalent” and $\delta(p, a)$ and $\delta(q, a)$ are marked as “not equivalent” for some $a \in \Sigma$, mark p and q as “not equivalent.”
3. Repeat the previous step until no more pairs of states need to be marked as “not equivalent.”
4. Merge sets of “equivalent” states into individual states of the resulting automaton (see example).

Theorem: States p and q are marked as “not equivalent” iff there is a string that distinguishes them.

Proof: omitted.

Table Filling Algorithm

Build a 2D triangular Boolean array to represent whether each state is equivalent.

Mark more and more states as inequivalent until no more are marked.

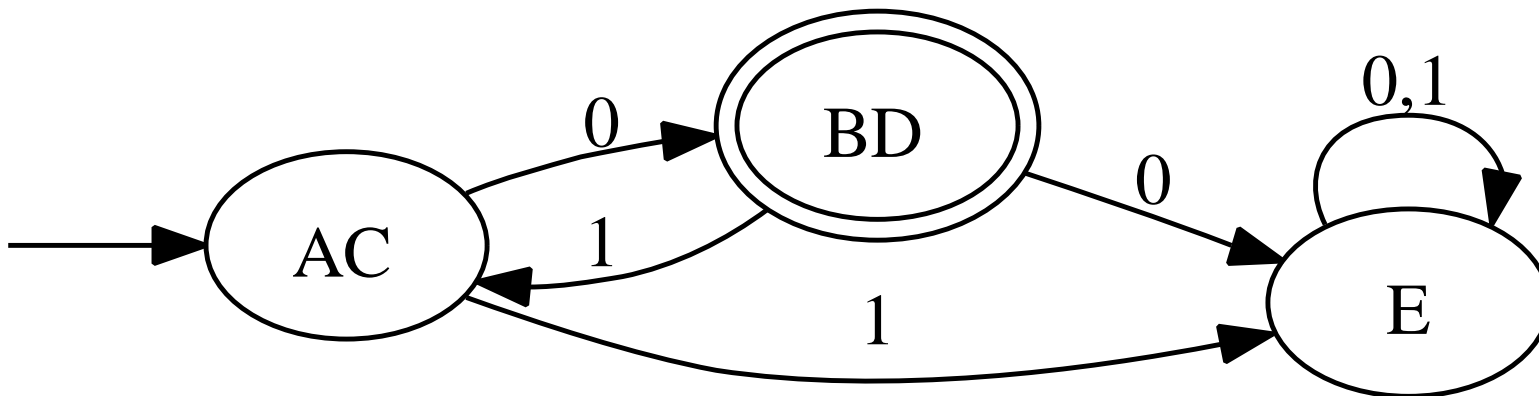
B	x			
C		x		
D	x		x	
E	x	x	x	x
	A	B	C	D

This terminates because there are finitely many table entries.

$A = C, B = D, E.$

Minimization example

To minimize, merge equivalent states.



Detail: Also should delete states that are not reachable from q_0 .

The resulting DFA is the unique minimum-state automaton.

It is the same as every other minimum-state DFA, except for state naming.

Some decision problems

Decision problem: A question with a yes/no answer.

Is string $x \in L(N)$? (The membership problem.)

Solution: Compute $\hat{\delta}(q_0, x)$, check for final states.

Given a finite automaton, N , is $L(N) = \emptyset$? (The emptiness problem.)

Given a FA, how do you check for emptiness reasonably quickly?

Answer: Do depth-first or breadth-first search from q_0 for a reachable final state.

Is $L(N) = \Sigma^*$? (The universality problem.)

Answer: Check $\overline{L(N)} = \emptyset$. This is easy if N is a DFA. Otherwise, your best bet is to convert it to a DFA and complement.

Is $L(N) \subseteq L(M)$? (The subset problem)

Answer: Check for $L(N) \cap \overline{L(M)} = \emptyset$

Equivalence?

Answer: Check for subset in both directions, OR minimize both DFAs and check if they are the same.