Regular Expressions and Language Properties

Mridul Aanjaneya

Stanford University

July 3, 2012

Tentative Schedule

- HW #1: Out (07/03), Due (07/11)
- HW #2: Out (07/10), Due (07/18)
- HW #3: Out (07/17), Due (07/25)
- Midterm: 07/31
- HW #4: Out (07/31), Due (08/08)
- Tentative grades out by 08/12.
- Final: ?

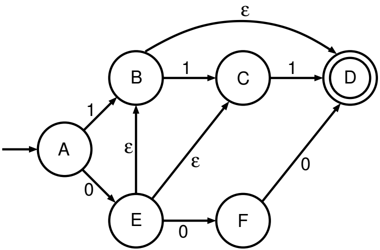
Epsilon Transitions: Extended transition function

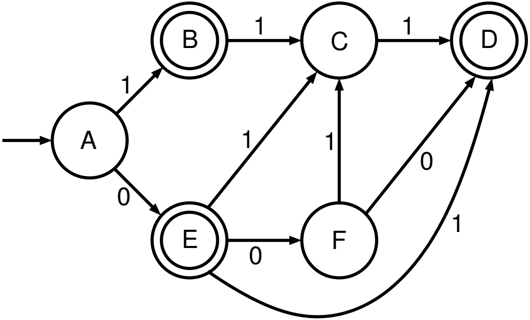
- **Basis:** δ_E(q,ε) = CL(q).
- **Induction:** δ_E(q,xa) is computed as follows:
  1. Start with δ_E(q,x) = S.
  2. Take the union of CL(δ(p,a)) for all p in S.
- **Intuition:** δ_E(q,w) is the set of states you can reach from q following a path labeled w with ε's in between.

Equivalence of NFA, ε-NFA

- Compute δ_N(q,a) as follows:
  - Let S = CL(q).
  - δ_N(q,a) is the union over all p in S of δ_E(p,a).
- F' = set of states q such that CL(q) contains a state of F.
- **Intuition:** δ_N incorporates ε-transitions before using a.
- Proof of equivalence is by induction on |w| that CL(δ_N(q_0,w)) = δ_E(q_0,w).
- **Basis:** CL(δ_N(q_0,ε)) = CL(q_0) = δ_E(q_0,ε).
- **Inductive step:** Assume **IH** is true for all x shorter than w. Let w = xa.
  - Then CL(δ_N(q_0,xa)) = CL(δ_E(CL(δ_N(q_0,x)),a)) (by definition).
  - But from **IH**, CL(δ_N(q_0,x)) = δ_E(q_0,x).
  - Hence, CL(δ_N(q_0,w)) = CL(δ_E(δ_E(q_0,x),a)) = δ_E(q_0,w).

Example





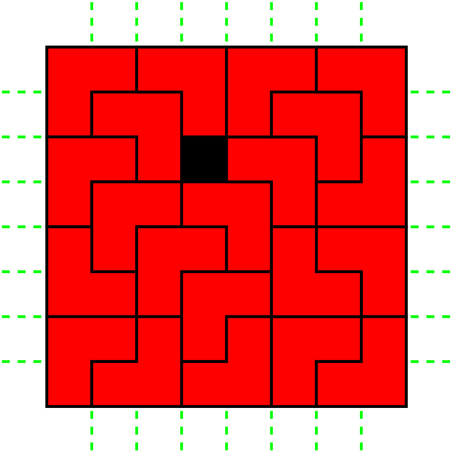
Recap

- DFA's, NFA's and ε-NFA's all accept exactly the same set of languages: the regular languages.
- NFA types are easier to design and may have exponentially fewer states than a DFA.
- But only a DFA can be implemented!

Challenge Problem #1

Question

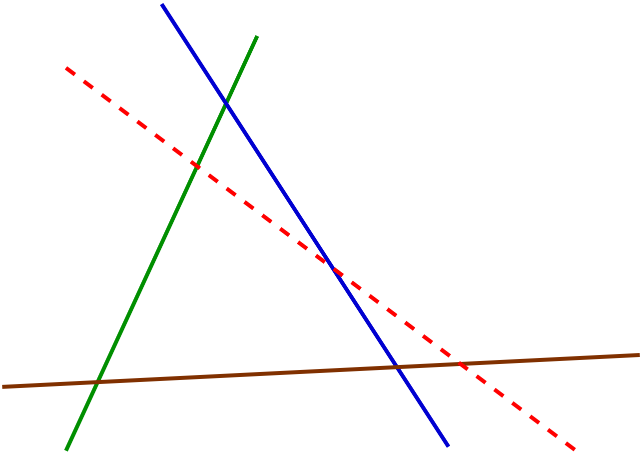
Are such tilings always possible?



Challenge Problem #2

Question

How many regions can you cut?



Regular Operators

- We define three regular operations on languages.

Definition

Let A and B be languages. We define the regular operations union, concatenation, and star as follows.

- **Union:** $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.
- **Concatenation:** $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$.
- **Star:** $A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$.

Kleene Closure

Denoted as A^* and defined as the set of strings $x_1 x_2 \ldots x_n$, for some $n \geq 0$, where each x_i is in A.

- **Note:** When $n = 0$, the string is ε.

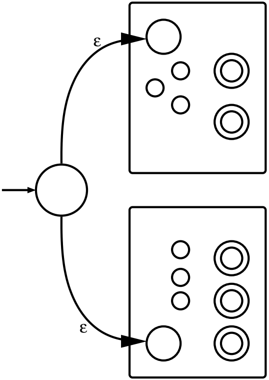
Example

- Let $\Sigma = \{a, b, \ldots, z\}$. If $A = \{\text{good}, \text{bad}\}$ and $B = \{\text{boy}, \text{girl}\}$,
- $A \cup B = \{\text{good}, \text{bad}, \text{boy}, \text{girl}\}$,
- $A \circ B = \{\text{goodboy}, \text{goodgirl}, \text{badboy}, \text{badgirl}\}$,
- $A^* = \{\varepsilon, \text{good}, \text{bad}, \text{goodgood}, \text{goodbad}, \text{badgood}, \text{badbad}, \ldots\}$,

Closure Properties: Union

Theorem

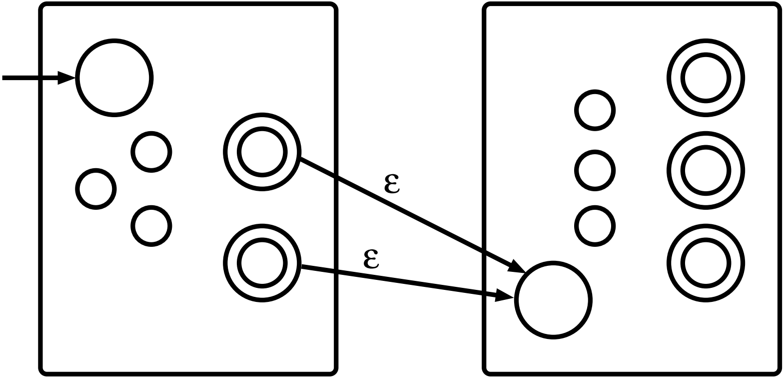
The class of regular languages is closed under the union operation, i.e., if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.



Closure Properties: Concatenation

Theorem

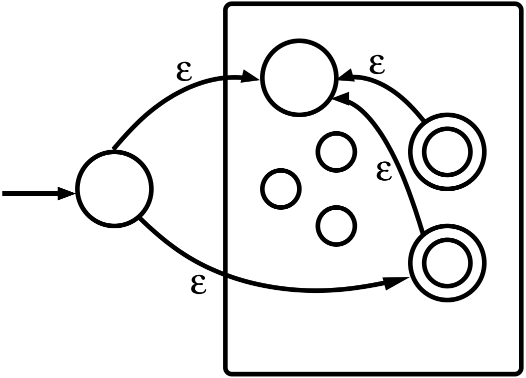
The class of regular languages is closed under concatenation.



Closure Properties: Star

Theorem

The class of regular languages is closed under the star operation.



Regular Expressions

- Regular expressions describe languages algebraically.
- They describe exactly the regular languages.
- If E is a regular expression, then $L(E)$ is its language.
- We give a recursive definition of RE's and their languages.

Regular Expressions: Definition

1. **Basis:** If a is any symbol, then a is a RE, and L(a)={a}.
   - **Note:** {a} is the language containing one string, and that string is of length 1.
2. **Basis:** ε is a RE, and $L(\varepsilon) = \{\varepsilon\}$.
3. **Basis:** $\emptyset$ is a RE, and $L(\emptyset) = \emptyset$.

Regular Expressions: Definition

1. **Induction:** If E_1 and E_2 are regular expressions, then E_1+E_2 is a regular expression, and $L(E_1+E_2) = L(E_1) \cup L(E_2)$.
2. **Induction:** If E_1 and E_2 are regular expressions, then E_1E_2 is a regular expression, and $L(E_1E_2) = L(E_1)L(E_2)$.
3. **Induction:** If E is a regular expression, then E^* is a regular expression, and $L(E^*) = (L(E))^*$.

Precedence of operators

- Parentheses may be used wherever needed to influence the grouping of operators.
- Order of precedence is * (highest), then concatenation, then + (lowest).

Examples

- $L(01) = \{01\}$.
- $L(01+0) = \{01,0\}$.
- $L(0(1+0)) = \{01,00\}$.
  - **Note:** order of precedence.
- $L(0^*) = \{\varepsilon,0,00,000,\ldots\}$
- $L((0+10)^*(\varepsilon+1))$ = all strings over $\{0,1\}$ without 11's.

Algebraic Laws for Regular Expressions

- Union and concatenation behave sort of like addition and multiplication.
- $+$ is commutative and associative.
- concatenation is associative.
- concatenation distributes over $+$.
- **Exception:** concatenation is not commutative.

Identities and Annihilators

- $\emptyset$ is the identity for $+$.
  - $R + \emptyset = R$.
- ε is the identity for concatenation.
  - $\varepsilon R = R\varepsilon = R$
- $\emptyset$ is the annihilator for concatenation.
  - $\emptyset R = R\emptyset = \emptyset$.

Equivalence of Regular Expressions and Automata

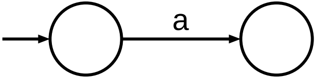
- We need to show that for every regular expression, there is an automaton that accepts the same language.
  - Pick the most powerful automaton type: ε-NFA.
- And we need to show that for every automaton, there is a regular expression defining its language.
  - Pick the most restrictive type: the DFA.

Converting a RE to an ε-NFA

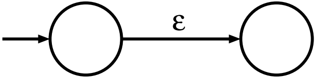
- Proof is an induction on the number of operators ($+$, concatenation, *) in the regular expression.
- We always construct an automaton of a special form (next slide).

RE to ε-NFA: Basis

- Symbol a:



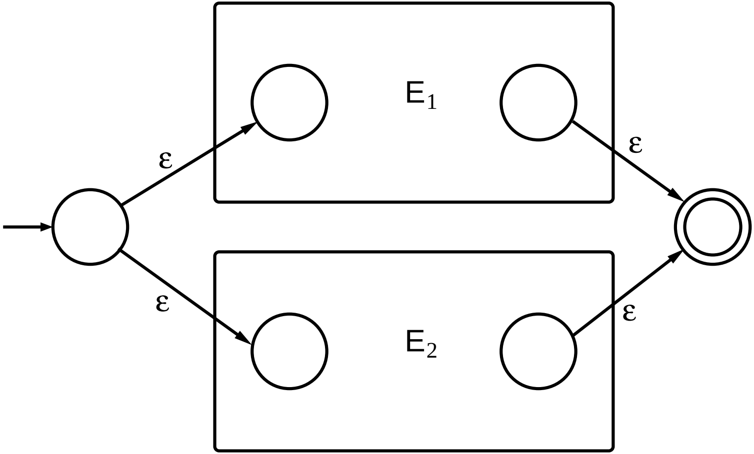
- ε:



- $\emptyset$:

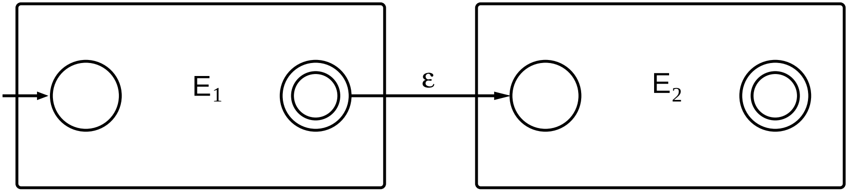
RE to ε-NFA: Induction (Union)

- For $E_1 \cup E_2$



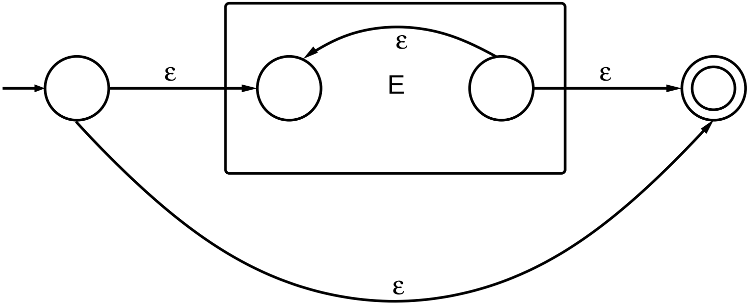
RE to ε-NFA: Induction (Concatenation)

- For E_1E_2



RE to ε-NFA: Induction (Star)

- For E*



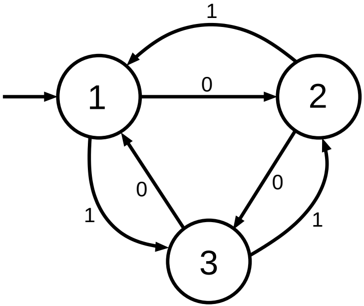
DFA to RE

- A strange sort of induction.
- States of the DFA are assumed to be $1, 2, \ldots, n$.
- We construct RE's for the labels of restricted sets of paths.
  - **Basis:** single arcs or no arcs at all.
  - **Induction:** paths that are allowed to traverse next state in order.

k-Paths

- A k-path is a path through the DFA that goes through no state numbered higher than k.
- End-points are not restricted, they can be any state.

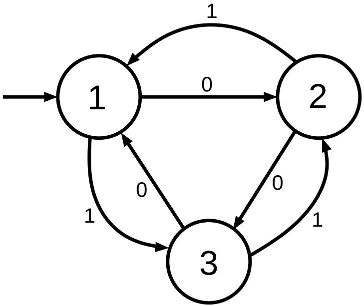
k-Paths: Example



- 0-paths from 2 to 3: RE for labels = 0
- 1-paths from 2 to 3: RE for labels = 0+11
- 2-paths from 2 to 3: RE for labels = $(10)^*0+1(01)^*1$
- 3-paths from 2 to 3: RE for labels = ??

k-Paths: Induction

- Let R_{ij}^{k} be the RE for the set of labels of k-paths from state i to state j.
- **Basis:** $k = 0$. R_{ij}^{0} = sum of labels of arcs from i to j.
  - $\emptyset$ is no such arc.
  - But add ε if i=j.



- **Example:** $R_{12}^{0} = 0$, $R_{11}^{0} = \emptyset + \varepsilon = \varepsilon$.

k-Paths: Inductive Step

- A k-path from i to j either:
  1. Never goes through state k, or
  2. Goes through state k one or more times.

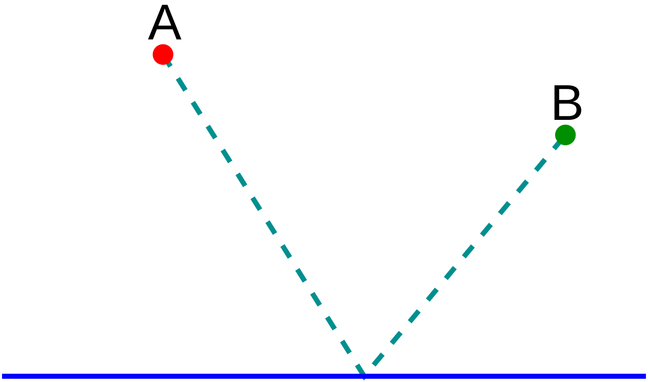
  $R_{ij}^{k} = R_{ij}^{k-1} + R_{ik}^{k-1}(R_{kk}^{k-1})^{*}R_{kj}^{k-1}$
- The equivalent RE is the sum (union) of R_{ij}^{n}, where:
  1. n is the number of states, i.e., the paths are unconstrained.
  2. i is the start state.
  3. j is one of the final states.

Summary

- Each of the three types of automata (DFA, NFA, ε-NFA) we discussed, and regular expressions as well, define exactly the same set of languages: the regular languages.

Question

Can you find the shortest path from A to B?



Properties of Language Classes

- A language class is a set of languages.
  - We have seen one example: the regular languages.
  - We'll see many more in the class.
- Language classes have two important kinds of properties:
  1. Decision properties
  2. Closure properties

Representation of Languages

- Representations can be formal or informal.
- **Example** (formal): represent a language by a DFA or RE defining it.
- **Example** (informal): a logical or prose statement about its strings:
  - $\{0^n 1^n | n$ is a nonnegative integer.$\}$
  - The set of strings consisting of some number of 0's followed by the same number of 1's.

Decision Properties

- A decision property for a class of languages is an algorithm that takes a formal description of a language (e.g., a DFA) and tells whether or not some property holds.
- **Example:** Is language L empty?

Representation matters ...

- You might imagine that the language is described informally, so if my description is the empty language then yes, otherwise no.
- But the representation is a DFA (or a RE that you will convert to a DFA).
- Can you tell if $L(A) = \emptyset$ for a DFA A?

Why Decision Properties?

- Remember that DFA's can represent protocols, and good protocols are related to the language of the DFA.
- **Example:** Does the protocol terminate? = Is the language finite?
- **Example:** Can the protocol fail? = Is the language nonempty?
- We might want a smallest representation for a language, e.g., a minimum-state DFA or a shortest RE.
- If you can't decide "Are these two languages the same?", i.e., do two DFA's define the same language - you can't find a "smallest"!

Closure Properties

- A closure property of a language class says that given languages in the class, an operator (e.g., union) produces another language in the same class.
- **Example:** We saw that regular languages are closed under union, concatenation and Kleene closure (star) operations.

Why Closure Properties?

- Helps construct representations.
- Helps show (informally described) languages not to be in the class.

The Membership Question

- Our first decision property is the question: "is the string w in regular language L?"
- Assume L is represented by a DFA A.
- Simulate the action of A on the sequence of input symbols forming w.

Question

What if L is not represented by a DFA?

- Use the circle of conversions:

$$\text{RE} \rightarrow \varepsilon\text{-NFA} \rightarrow \text{NFA} \rightarrow \text{DFA} \rightarrow \text{RE}$$

The Emptiness Problem

Question

Does a regular language L contain any string at all?

- Assume representation is a DFA.
- Compute the set of states reachable from the start state.
- If any final state is reachable, then yes, else no.

The Infiniteness Problem

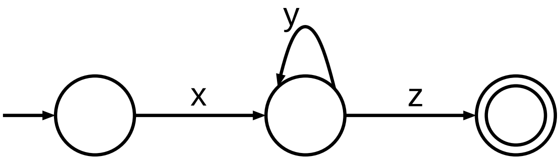
Question

Is a given regular language L infinite?

- Start with a DFA for the language.
- **Key idea:** If the DFA has n states, and the language contains any string of length n or more, then the language is infinite.
- Otherwise, the language is surely finite.
  - Limited to strings of length n or less.

Proof of Key Idea

- If an n-state DFA accepts a string w of length n or more, then there must be a state that appears twice on the path labeled w from the start state to a final state.
  - **Note:** Pigeonhole principle! $\ddot{\smile}$
- Because there are at least $n + 1$ states along the path.



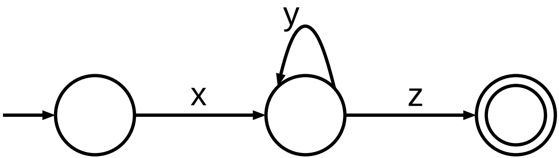
- Since y is not ε, we see an infinite number of strings in L of the form xy^iz for all $i \geq 0$.

The Infiniteness Problem

- We do not have an algorithm yet.
- There are an infinite number of strings of length $> n$, and we can't test them all!
- **Second Key Idea:** If there is a string of length $\geq n$, then there is a string of length between n and $2n - 1$.

Proof of Second Key Idea

- Remember:



- We can choose y to be the first cycle on the path.
- So $|xy| \leq n$; in particular, $1 \leq |y| \leq n$.
- Thus, if w is of length $2n$ or more, there is a shorter string in L that is still of length at least n.
- Keep shortening to reach $[n, 2n - 1]$.

Completion of Infiniteness Algorithm

- Test for membership all strings of length between $[n, 2n-1]$.
  - If any are accepted, then infinite, else finite.
- A terrible algorithm!
- **Better:** find cycles between the start state and a final state.
- For finding cycles:
  1. Eliminate states not reachable from the start state.
  2. Eliminate states that do not reach a final state.
  3. Test if the remaining transition graph has any cycles.