CS375:
Logic and Theory of Computing

Fuhua (Frank) Cheng

Department of Computer Science

University of Kentucky

Table of Contents:

- **Week 1: Preliminaries (set algebra, relations, functions) (read Chapters 1-4)**
- **Weeks 2-5: Regular Languages, Finite Automata (Chapter 11)**
- **Weeks 6-8: Context-Free Languages, Pushdown Automata (Chapters 12)**
- **Weeks 9-11: Turing Machines (Chapter 13)**

Table of Contents (conti):

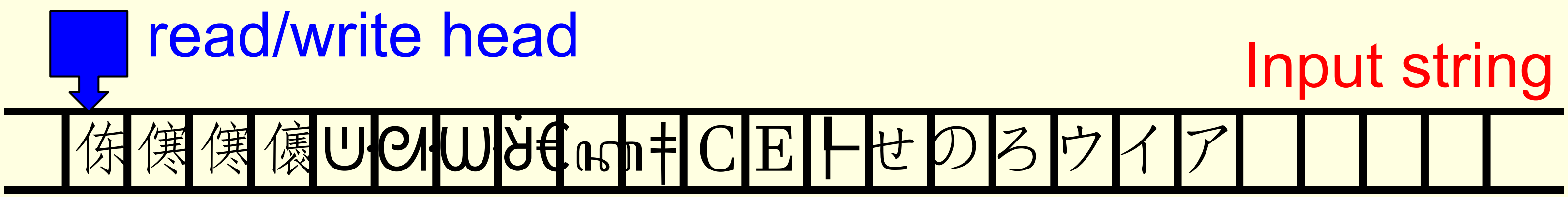
- **Weeks 12-13: Propositional Logic (Chapter 6), Predicate Logic (Chapter 7), Computational Logic (Chapter 9), Algebraic Structures (Chapter 10)**

Regular Languages & Finite Automata

- Regular Languages

Goal:

Try to answer the question: "can a machine recognize a language?"



If the answer is YES, then what kind of languages can be recognized by what kind of machines?

A smart machine

Remember these:

Discrete, *not continous*

First, the language must be a DIGITAL language (Otherwise, the language has to be DIGITIZED).

Second, the machine must be a DIGITAL machine, i.e., the machine must use a digital process to read/execute the language.

No digital process, No computer!

Regular Languag

A *description*, not a *representation*.
We need something like: *a*bba**

Problem:

Suppose the input strings strings over the alphabet *{a, b}* that conta s exactly one substring *bb*, i.e., the strings are of the form

xbby

where *x*, *y*: strings over *{a, b}* that do not contain *bb*, *x* does not end in *b*, and *y* does not begin with *b*.

Question: how to **represent** the strings formally?

Remember these:

For us, not for the machine.

Description : a set of statements

Representation : a form that carries a specific internal structure. E.g. x*

So, is 'xbby' a representation?

Yes, but not precise enough. Why?

*Can you tell the **internal structure** of x by looking at it?*

Regular Languages:

set of strings over A

A ***regular language*** over alphabet A is a language constructed by the following rules:

Means you can have single-letter words

- $\emptyset$ and $\{\Lambda\}$ are regular languages.
- $\{a\}$ *is a regular language for all* $a \in A$.
- If M and N are regular languages, then so are $M \cup N$, MN, and M^*.

Means you can have multiple-word/sentence languages

Means you can have multiple-letter words and sentences

Means you can have everything above

closure of M: all possible concatenations of strings from M

Regular Languages:

set of strings over A

A ***regular language*** over alphabet A is a language constructed by the following rules:

Basis

- $\emptyset$ and $\{\Lambda\}$ are regular languages.
- *$\{a\}$ is a regular language for all $a \in A$.*

- If M and N are regular languages, then so are $M \cup N$, MN, and M^*.

Induction

$$M^* = \bigcup_{i=0}^{\infty} M^i$$

Why do we need an empty set and an empty string?

A set with three elements has how many subsets?

Eight (one of them is the empty set)

A string with three letters has how many substrings?

Eight (one of them is the empty string)

Regular Languages:

Example. Regular languages over $A = \{a, b\}$:

$\emptyset$, $\{\Lambda\}$, $\{a\}$, $\{b\}$, $\{a, b\}$, $\{ab\}$, $\{a\}^* = \{\Lambda, a, aa, aaa, \ldots\}$

A ***regular expression*** over A is an expression constructed by the following rules:

representation

- $\emptyset$ and Λ are regular expressions.
- a is a regular expression for all $a \in A$.
- If R and S are regular expressions, then so are

  (R), $R + S$, $R \cdot S$, and R^*.

do R first

R or S

Remember these:

Language = set of strings

Expression
= (format) representation
= general representation
= symbolic representation

Set vs Expression

Set = UK employees and students

Expression (format) :

Name	UK ID number	Date of birth

Set element:

John Smith	912373436	20050606

Set vs Expression

The set of polynomials in the variable x

= $\{2+3x-11x^2,\ 49+6x^3+5x^7, \ldots \qquad \}$

Or

= $\{\, f(x) \mid f(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$ *for some non-negative integer* n *and* $a_0, a_1, a_2, \ldots, a_n$ *are real numbers* $\}$

Set vs Expression

$f(x) + g(x) = ?$ $f(x)\, g(x) = ?$ $f(x)^* = ?$

You can use real polynomials to show the addition, multiplication, subtraction … of polynomials.

But it is more general to use symbolic representation to explain the addition, multiplication, subtraction … process.

Use symbolic representation to define addition, …

If $f(x) = a_0 + a_1 x + \cdots + a_n x^n$

$g(x) = b_0 + b_1 x + \cdots + b_n x^n + \cdots + b_m x^m$

and m ≥ n, then f(x) + g(x) = ?

Convert f(x) to a degree m polynomial and then perform coefficient addition for terms with the same exponent.

So, why do we want to study regular expressions?

Because dealing with languages (sets) directly is a very time consuming process.

Note that most of the practical regular languages are infinite sets. Dealing with infinite sets directly means we have to list those sets explicitly, a very time consuming (and actually impossible) process.

On the other hand, if we deal with regular expressions of regular languages instead, we only have to list a few representations (format or symbolic descriptions), a much simpler process.

Regular Languages:

Another way to understand expression:

(1) Φ is a regular expression of the empty language.

(2) Λ is a regular expression of {Λ}.

(3) For any symbol a, a is a regular expression of {a}.

(4) If R_A and R_B are regular expressions of languages A and B, then R_A+R_B is a regular expression of A U B, R_AR_B is a regular expression of AB, and $R_A{}^*$ is a regular expression of A*.

Regular Languages:

Not a regular expression of {0}{1}

Examples:

- **011** : regular expression of {0}{1}{1}.
- **0+1** : regular expression of {0,1}.
- **(0+1)*** : regular expression of {0,1}*.
- Remark: **(0+1)** is also considered a regular expression of {0, 1}.

Regular Languages:

The hierarchy in the absence of parentheses is:

$*$, $\cdot$, $+$

$$a + b \cdot a^* = (a + (b \cdot (a)^*)) \equiv a + ba^*$$

Juxtaposition will be used in place of $\cdot$.

Example. The following expressions are a sampling of the regular expressions over A = {a, b}:

∅, Λ, a, b, ab, a + ab, (a + b).*

$$\begin{cases} \Lambda : \text{expression} \\ \{\Lambda\} : \text{language} \end{cases}$$

Regular Languages:

expression *set* *expression* *element*

Regular expressions vs regular languages

by the following correspondence, where $L(R)$ denotes the regular language of the regular expression R.

$L(\emptyset) = \emptyset$,

$L(\Lambda) = \{\Lambda\}$,

$L(a) = \{a\}$ for all $a \in A$,

$L(R + S) = L(R) \cup L(S)$,

$L(RS) = L(R)L(S)$,

$L(R^*) = L(R)^*$

Regular Languages:

Example. $L(ab + a^*)$ represents the following regular language:

$$
\begin{aligned}
L(ab + a^*) &= L(ab) \cup L(a^*) \\
&= L(a)L(b) \cup L(a)^* \\
&= \{a\}\{b\} \cup \{a\}^* \\
&= \{ab\} \cup \{\Lambda, a, aa, aaa, \ldots, a^n, \ldots\} \\
&= \{ab, \Lambda, a, aa, aaa, \ldots, a^n, \ldots\}.
\end{aligned}
$$

$L(R + S) = L(R) \cup L(S)$

$L(RS) = L(R)L(S)$, $\boxed{L(R^*) = L(R)^*}$

$L(a) = \{a\}$

Example. $L((a + b)^*)$ represents the following regular language:

$L((a + b)^*) = (L(a + b))^* = (L(a) \cup L(b))^* = (\{a\} \cup \{b\})^* = \{a, b\}^*$

set of all possible strings over {a, b}

Is this statement true?

A language can be represented by a regular expression if and only if that language is a regular language.

Question: *can you find a regular expression for* $\{ a^n b^n \mid n \in N \}$?

First of all, is $B = \{ a^n b^n \mid n \in N \}$ *regular?* **No**

And is $A = \{ a^n \mid n \in N \}$ *regular?* **Yes**

Why?

$A = \{a^n \mid n \in N\}$
$= \{\Lambda, a, aa, aaa, \ldots, a^n, \ldots\}$
So A = {a}*.

$B = \{a^n b^n \mid n \in N\}$
$= \{\Lambda, ab, aabb, \ldots, a^n b^n, \ldots\}$
$\neq \{ab\}^*$
$\neq \{a\}^*\{b\}^*$
Is there a regular expression for B?

Joke for today:

Me: The new girl in our neighborhood smiled at me today!

Wife: Be careful, she has Covid.

Me: What? How do you know?

Wife: Can’t you see she has no taste?

So, what conclusion can you draw here?

The wife has no taste either.

Regular Languages:

Any possible combinations of strings from a and (ba)**

Back to the Problem: Suppose input strings are strings over the alphabet *{a, b}* that contain exactly one substring *bb*. That is, the strings must be of the form

xbby

where ***x*** and ***y*** are strings over *{a, b}* that do not contain *bb*, ***x*** does not end in *b*, and ***y*** does not begin with *b*.

How can we describe the set of these strings formally?

Solution: let $x = (a + ba)^*$ and $y = (a + ab)^*$.

$$L((a+ba)^*) = (L(a+ba))^* = (L(a) \cup L(ba))^*$$

$$= (\{a\} \cup \{ba\})^* = \{a, ba\}^*$$

$$= \{\Lambda, a, ba, aa, aaa, \cdots, a^n, \cdots, baba, \cdots, (ba)^n, \cdots, aba, \cdots\}$$

Write these down:

x could be Φ or Λ

A : a set

$$A^* \equiv A^0 \cup A^1 \cup A^2 \cup A^3 \cup \cdots \cup A^n \cup \cdots$$

$$\equiv \emptyset \cup A^1 \cup A^2 \cup A^3 \cup \cdots \cup A^n \cup \cdots$$

x : a string (could be a single-letter string)

$$x^* \equiv x^0 \cup x^1 \cup x^2 \cup x^3 \cup \cdots \cup x^n \cup \cdots$$

$$\equiv \Lambda \cup x^1 \cup x^2 \cup x^3 \cup \cdots \cup x^n \cup \cdots$$

Regular Languages:

Back to the Problem: *………………………..*

……….

where *x* and *y* are strings over *{a, b}* that do not contain *bb*, *x* does not end in *b*, and *y* does not begin with *b*.

How do I know that x = (a + ba)* ?

x = a   x = a a b a   x = a a a b a a a b a

x = a a a a b a a a a b a a a a b a b a b a

So each x is composed of elements from {a}* or/and {ba}*.
(note that x could be equal to Λ)

Regular Languages:

Question:

x = (ba)*   ?

No, because it does not cover strings like
a, aba, ababa, …..

or

aa, aaa, aaaa, …..

Regular Languages:

Question:

x = (aba)* ?

No, why?

abaaba

abaabaaba

L((aba)*)={Λ, $(aba)^1$, $(aba)^2$, $(aba)^3$, …, $(aba)^n$, …}

does not cover strings like a, aa, …, ba, baa, …, ababa, abaaaba, …..

Regular Languages:

Quiz. Find a regular expression for

$$\{\ ab^n \mid n \in \mathbf{N}\} \cup \{\ ba^n \mid n \in \mathbf{N}\}.$$

Answer. *ab* + ba**

Quiz. Use a sentence to describe the language of

$$(b + ab)^*(\Lambda + a).$$

Answer. All strings over *{a, b}* whose substrings of *a*'s have length 1.

(or, all strings over {a, b} that do not contain the substring aa)

Regular Languages:

Quiz. Find a regular expression for

$$\boxed{\{\, ab^n \mid n \in \mathbf{N} \} \cup \{\, ba^n \mid n \in \mathbf{N} \}.}$$

Answer. $\boxed{ab^* + ba^*}$ *Why?*

$L(ab^* + ba^*) = L(ab^*) + L(ba^*)$

$= L(a)\, L(b^*) + L(b)\, L(a^*)$

$= L(a)\, L(b)^* + L(b)\, L(a)^*$

$= \{a\}\, \{b\}^* \; U \; \{b\}\, \{a\}^*$

$= \{\, ab^n \mid n \in N \} \cup \{ba^n \mid n \in N \}$

Regular Languages:

Quiz. Use a sentence to describe the language of

$(b + ab)^*(\Lambda + a)$.

Answer. *All strings over {a, b} whose substrings of a's have length 1*

(or, all strings over {a, b} that do not contain the substring aa)

Why?

$= (b + ab)^*\Lambda + (b + ab)^*a$ ***intuitively***

$= \underline{(b + ab)^*} + \underline{(b + ab)^*a}$

$b^*(ab)^*\, b^*(ab)^* \ldots \; \Lambda$

$b^*(ab)^* b^*(ab)^* \ldots \; a$

Strings in the language of $(b + ab)^*(\Lambda + a)$:

$$(b + ab)^*(\Lambda + a) = \underline{(b + ab)^*\Lambda} + \underline{(b + ab)^*a}$$

b	ba
ab	aba
bab	baba
bbab	bbaba
bbabab	bbababa
.........b	.........ba

Skip the next three slides

Reg[...]ges:

How should this part be modified?

How should this part be modified?

We just proved [...]at:

The following is an [...]xpression fo[...]ll strings over $\{a, b\}$ that do not contain the s[...]bstring aa:

$$(b + ab)^* (\Lambda + a)$$

Question:

What is an expression for all strings over $\{a, b\}$ that do not contain the substring aaa ?

Examples : babaab babaaba babaabaa

Answer: $(b + ab + aab)^*(\Lambda + a + aa)$

Regular Languages:

Known Results:

1. regular expression for strings over *{a, b*} with exactly one "*a*" ?

b^*ab^*

2. exactly 2 *a*'s ?

$b^*ab^*ab^*$

exponent of b could be 0

3. exactly 3 *a*'s ?

$b^*ab^*ab^*ab^*$

. . .

Regular Languages:

Is this part necessary?

Known Results:

4. regular expression for strings over {*a, b*} with even number of *a*'s ?

Answer: $\boxed{(b^*ab^*ab^*)^*}\,b^*$

$(b^*ab^*ab^*)^*b^*$

Is this part necessary?

$(b^*ab^*ab^*)^*$

covers Λ, $(b^*ab^*ab^*)$, $(b^*ab^*ab^*)^2$, $(b^*ab^*ab^*)^3$, …

but not b^*

such as b, bb, bbb, …

These strings also satisfy the requirement.

Regular Languages:

Known Results:

4. regular expression for strings over *{a, b}* with even number of *a*'s ?

$(b^*ab^*ab^*)^*b^*$ (or, $b^*(b^*ab^*ab^*)$)

Question:

What is a regular expression for strings over *{a, b}* with odd number of *a*'s ?

Would $(b^*ab^*)^*$ work?

No

What is a regular expression for strings over {*a*, *b*} with odd number of *a*'s ?

Would $(b^*ab^*ab^*)^*ab^*$ *work?*

Is bab or bbab covered by this regular expression?

Would $(b^*ab^*ab^*)^*b^*ab^*$ *work?*

YES

Or, $b^*(b^*ab^*ab^*)^*ab^*$ *?*

Algebra of Regular Expressions:

Equality: If L(R) = L(S), we say regular expressions *R* and *S* are equal and write *R* = *S*

Examples. $a + b = b + a$, $a + a = a$

L(a+b) = {a, b} = {b, a} = L(b+a)

L(a+a) = {a, a} = {a} = L(a)

Why do we want to study regular expression algebra?

Because studying relations between regular languages directly is a very complex and big job.

Most regular languages are infinite sets. Studying relations between regular languages directly means we have to deal with union, intersection, concatenation, … of infinite sets. Big job.

On the other hand, if we instead study relations between regular expressions, we only have to deal with addition, concatenation, … , of a few representations (formulas), a much smaller job.

Algebra of Regular Expressions:

Remember: $a^* \equiv \Lambda + a + a^2 + a^3 + \cdots$

Equality: If L(R) = L(S), we say Regular expressions *R* and *S* are equal and write *R* = *S*

Examples. $aa^* = a^*a$ $ab \neq ba$

L(aa*) = L(a) L(a)* = {a}{a}* = $\{a^i \mid i \in N^+\}$

= {a}*{a} = L(a)* L(a) = L(a*a)

L(ab) = L(a)L(b) = {a}{b} = {ab}

≠ {ba} = {b}{a} = L(b)L(a) = L(ba)

Algebra of Regular Expressions:

Properties of + , · , and closure

+ is commutative, associative,
  Φ is identity for +, and *R* + *R* = *R*.

$$\Phi + R = R$$
$$\Lambda + R \neq R$$

· is associative, Λ is identity for · ,
  and Φ is zero for · .

$$\Lambda R = R\Lambda = R$$

$$\Phi R = R\Phi = \Phi$$

· distributes over +

$$R(S+T) = RS + RT$$

Algebra of Regular Expressions:

However, $RR \neq R$

Properties of + :

$R+T = T+R$ $\quad L(R+T) = L(R) \cup L(T) = L(T) \cup L(R) = L(T+R)$

$R+\phi = \phi + R = R$ $\quad L(R+\Phi) = L(R) \cup L(\Phi) = L(R) \cup \Phi = L(R)$

$R+R = R$ $\quad L(R+R) = L(R) \cup L(R) = L(R)$

$(R+S)+T = R+(S+T)$

$$\begin{aligned} L((R+S)+T) &= L(R+S) \cup L(T) \\ &= ((L(R) \cup L(S)) \cup L(T) \\ &= L(R) \cup (L(S) \cup L(T)) \\ &= L(R+(S+T)) \end{aligned}$$

Algebra of Reg...ions:

Needed in slide 53 (number at lower right corner)

Properties of · :

$$\boxed{R\phi = \phi R = \phi} \qquad L(R\phi) = L(R)L(\phi) = \underline{L(R)\phi = \phi} = L(\phi)$$

$$\boxed{R\Lambda = \Lambda R = R} \qquad L(R\Lambda) = L(R)L(\Lambda) = \underline{L(R)\{\Lambda\} = L(R)}$$

$$\boxed{(RS)T = R(ST)} \qquad L((RS)T) = L(RS)L(T) = \underline{(L(R)L(S))\,L(T)}$$

$$= \underline{L(R)\,(L(S)L(T))}$$

$$= L(R)L(ST)$$

$$= L(R(ST))$$

Algebra of Regular Expressions:

Distributive properties:

$\boxed{R(S+T) = RS + RT}$

$$L(R(S+T)) = L(R)L(S+T)$$
$$= L(R)(L(S) \cup L(T))$$
$$= L(R)L(S) \cup L(R)L(T)$$
$$= L(RS) \cup L(RT)$$
$$= L(RS + RT)$$

Algebra of Regular Expressions:

Closure Properties:

Needed in slide 54 (number at lower right corner)

- $\emptyset^* = \Lambda^* = \Lambda$.
- $R^* = R^*R^* = (R^*)^* = R + R^*$. (*)
- $R^* = \Lambda + R^* = \Lambda + RR^* = (\Lambda + R)^* = (\Lambda + R)R^*$.
- $R^* = (R + R^2 + \ldots + R^k)^* = \Lambda + R + R^2 + \ldots + R^{k-1} + R^k R^*$ for any $k \geq 1$.
- $R^*R = RR^*$. (**)
- $(R + S)^* = (R^* + S^*)^* = (R^*S^*)^* = (R^*S)^*R^* = R^*(SR^*)^*$. (***)
- $R(SR)^* = (RS)^*R$. (****)
- $(R^*S)^* = \Lambda + (R + S)^*S$ and $(RS^*)^* = \Lambda + R(R + S)^*$.
- $(R+S)^*S^* = (R+S)^*$ (*****)

Needed in slide 54 (number at lower right corner)

Algebra of Regular Expressions:

Closure properties:

- $\emptyset^* = \Lambda^* = \Lambda$.

Skip the next 2 slides

Proof: $\Phi^* \equiv \Phi^0 \cup \Phi^1 \cup \Phi^2 \cup \cdots$

$$= \Lambda \cup \Phi \cup \Phi \cup \cdots = \Lambda$$

$$\Lambda^* \equiv \Lambda^0 \cup \Lambda^1 \cup \Lambda^2 \cup \cdots$$

$$= \Lambda \cup \Lambda \cup \Lambda \cup \cdots = \Lambda$$

Algebra of Regular Expres

Needed in slide 54 (number at lower right corner)

Closure properties:

- $R^* = \Lambda + R^* = \Lambda + RR^* = (\Lambda + R)^* = (\Lambda + R)R^*$

Proof *of* $R^* = \Lambda + RR^*$.

$$L(\Lambda + RR^*) = L(\Lambda) \cup L(R)L(R)^* = \{\Lambda\} \cup L(R)^1 \cup L(R)^2 \cup \cdots$$
$$= L(R)^*.$$

Proof *of* $R^* = (\Lambda + R)R^*$.

$$L((\Lambda + R)R^*) = L(R^* + RR^*) = L(R)^* \cup L(R)L(R)^*$$
$$= L(R)^* \cup L(R)^+ = L(R)^*.$$

Skip the next 2 slides

Algebra of Regular Expressions:

$$L(R)L(R)^* = L(R)\,(L(R)^0 \cup L(R)^1 \cup L(R)^2 \cup \cdots)$$

$$= L(R)\,(\{\Lambda\} \cup L(R)^1 \cup L(R)^2 \cup \cdots)$$

$$= L(R)^1 \cup L(R)^2 \cup L(R)^3 \cdots$$

$$= L(R)^+$$

Algebra of Regular Expressions:

Closure properties:

- $R^* = \Lambda + R^* = \Lambda + RR^* = (\Lambda + R)^* = (\Lambda + R)R^*$

Proof of $R^* = (\Lambda + R)^*$.

Need to show that $L(R)^* = L(\Lambda + R)^* = (L(\Lambda) \cup L(R))^*$.

One direction is obvious. Since $L(R) \subseteq L(\Lambda) \cup L(R)$, we have $L(R)^* \subseteq (L(\Lambda) \cup L(R))^*$. On the other hand, if $x \in ((L(\Lambda) \cup L(R))^*$ then $x \in (L(\Lambda) \cup L(R))^m$ for some $m \in N$. Hence, for each $1 \le i \le m, \exists x_i \in L(\Lambda) \cup L(R)$ such that $x = x_1 x_2 \cdots x_m$. Without loss of generality, we can assume that $x_i \neq \Lambda$ for each i. But then $x \in L(R)^m \subseteq L(R)^*$. Hence, we also have $(L(\Lambda) \cup L(R))^* \subseteq L(R)^*$.

Algebra of Regular Expressions:

***Explain** each inequality.*

$(1).\ (a + b)^* \neq a^* + b^*.$ $(2)\ (a + b)^* \neq a^*b^*.$

Answers. (1) $ab \in LHS$, but not RHS

(2) $ba \in LHS$, but not RHS

Simplify regular expression $aa(b^* + a) + a(ab^* + aa).$

Answer. $aa(b^* + a) + a(ab^* + aa)$

$= aa(b^* + a) + aa(b^* + a)$ ← $\boxed{\cdot \text{ distributes over } +}$

$= aa(b^* + a)$ ← $\boxed{R + R = R}$

Algebra of Regular Expressions:

Example. *Show that* $(a + aa)(a + b)^* = a(a + b)^*$.

Proof I:

$(a + aa)\underline{(a + b)^*} = (a + aa)\underline{a^*(ba^*)^*}$ ← $(R + S)^* = R^*(SR^*)^*$

Slides 42 and 45

$= a\underline{(\Lambda + a)a^*}(ba^*)^*$ ← $R = R\Lambda$ and $\cdot$ distributes over $+$

$= \underline{aa^*(ba^*)^*}$ ← $(\Lambda + R)R^* = R^*$

$= a(a + b)^*$ ← $(R + S)^* = R^*(SR^*)^*$

QED.

Skip the next 9 slides

Example. *Show that*

$(a + aa + \ldots + a^n)(a + b)^* = a(a + b)^*$ for all $n \geq 1$.

Proof (by induction): If $n = 1$, the statement becomes

$a(a + b)^* = a(a + b)^*$, obviously true.

If $n = 2$, the statement becomes

$(a + aa)(a + b)^* = a(a + b)^*$, true by a previous example (slide 60).

Assume the statement is true for $1 \leq k < n$ $(n>2)$. We need to prove the statement is true for n.

Example. *Show that*

$$(a + aa + \dots + a^n)(a + b)^* = a(a + b)^* \qquad \text{for all } n \geq 1.$$

Proof (conti.): The LHS of the statement for *n is*

$$(a + aa + \dots + a^n)(a + b)^*$$

$$= a(a + b)^* + (aa + \dots + a^n)(a + b)^* \qquad \cdot \text{ distributes over } +$$

$$= a(a + b)^* + a(a + aa + \dots + a^{n-1})(a + b)^* \qquad \cdot \text{ distributes over } +$$

$$= a(a + b)^* + a\, a(a + b)^* \qquad \text{induction assumption}$$

$$= (a + aa)(a + b)^* \qquad \cdot \text{ distributes over } +$$

$$= a(a + b)^* \qquad \text{induction assumption}$$

$$= \text{RHS}$$

This is why n must be bigger than 2

End of Regular Language and Finite Automata I