COL 352 Introduction to Automata and Theory of Computation

Nikhil Balaji

Bharti 420
Indian Institute of Technology, Delhi
nbalaji@cse.iitd.ac.in

January 19, 2023

Lecture 7: Pattern Matching and Regular Expressions

Recap

- DFA = NFA = ε-NFA.
- All of them recognize (compute/decide) exactly regular languages
- Regular languages are closed under union, intersection, complement, concatenation, Kleene star, ...

A number of widely different and equiexpressive formalisms precisely capture the same class of languages:

Why regular?

- DFA = NFA = ε-NFA.
- All of them recognize (compute/decide) exactly regular languages
- Regular languages are closed under union, intersection, complement, concatenation, Kleene star, ...

A number of widely different and equiexpressive formalisms precisely capture the same class of languages:

- Type-3 languages in Chomsky's hierarchy.

Why regular?

- DFA = NFA = ε-NFA.
- All of them recognize (compute/decide) exactly regular languages
- Regular languages are closed under union, intersection, complement, concatenation, Kleene star, ...

A number of widely different and equiexpressive formalisms precisely capture the same class of languages:

- Type-3 languages in Chomsky's hierarchy.
- Monadic second-order logic definable languages (Buchi '60, Elgot '61, Trackhtenbrot'62)

Why regular?

- DFA = NFA = ε-NFA.
- All of them recognize (compute/decide) exactly regular languages
- Regular languages are closed under union, intersection, complement, concatenation, Kleene star, ...

A number of widely different and equiexpressive formalisms precisely capture the same class of languages:

- Type-3 languages in Chomsky's hierarchy.
- Monadic second-order logic definable languages (Buchi '60, Elgot '61, Trackhtenbrot'62)
- Certain Algebraic connection (acceptability via finite semi-group): Eilenberg'76

Why regular?

- DFA = NFA = ε-NFA.
- All of them recognize (compute/decide) exactly regular languages
- Regular languages are closed under union, intersection, complement, concatenation, Kleene star, ...

A number of widely different and equiexpressive formalisms precisely capture the same class of languages:

- Type-3 languages in Chomsky's hierarchy.
- Monadic second-order logic definable languages (Buchi '60, Elgot '61, Trackhtenbrot'62)
- Certain Algebraic connection (acceptability via finite semi-group): Eilenberg'76
- Regular expressions (Kleene'50s).
- Rational languages.

Algebraic view of regular languages

Consider any regular language

Algebraic view of regular languages

Consider any regular language

- It can be letters 1 or a or ϵ etc.
- It can be emptyset.
- It can be a word, e.g., 110: got as (finite) concatenation of letters.
- It can be several words: got as (finite) union of sets of words.

Algebraic view of regular languages

Consider any regular language

- It can be letters 1 or a or ϵ etc.
- It can be emptyset.
- It can be a word, e.g., 110: got as (finite) concatenation of letters.
- It can be several words: got as (finite) union of sets of words. Can these be used to generate all regular languages?
- It can be something that "repeats"!

Algebraic view of regular languages

Consider any regular language

- It can be letters 1 or a or ϵ etc.
- It can be emptyset.
- It can be a word, e.g., 110: got as (finite) concatenation of letters.
- It can be several words: got as (finite) union of sets of words. Can these be used to generate all regular languages?
- It can be something that "repeats"!

Kleene star

For a language L, its Kleene closure, denoted L^* is the set of all strings obtained by taking any number of strings from L with possible repetitions and concatenating all of them.

$$L^* = \cup_{i \geq 0} L^i$$

$$L^0 = \{\epsilon\}, L^i = L \circ L^{i-1}$$

Application: Pattern Matching

Application: Pattern Matching

- rm $*$.pdf
- ls, grep, awk etc.

Application: Pattern Matching

- rm $*$.pdf
- ls, grep, awk etc.
- Textual (declarative) way to represent regular languages (compared to automata).

Application: Pattern Matching

- rm $*$.pdf
- ls, grep, awk etc.
- Textual (declarative) way to represent regular languages (compared to automata).

Application: Pattern Matching

- rm $*$.pdf
- ls, grep, awk etc.
- Textual (declarative) way to represent regular languages (compared to automata).

Definition (Pattern)

A pattern α is a string of symbols of a certain form representing a (possibly infinite) set of strings in Σ^*.

Application: Pattern Matching

- rm $\ast$.pdf
- ls, grep, awk etc.
- Textual (declarative) way to represent regular languages (compared to automata).

Definition (Pattern)

A pattern α is a string of symbols of a certain form representing a (possibly infinite) set of strings in Σ^*.

$$L(\alpha) = \{x \in \Sigma^* \mid x \text{ matches } \alpha\}$$

Atomic Patterns

1. $a \in \Sigma,\ L(a) = \{a\}$

Atomic Patterns

1. $a \in \Sigma$, $L(a) = \{a\}$
2. ε, $L(\varepsilon) = \{\varepsilon\}$
3. $\varnothing$, $L(\varnothing) = \varnothing$
4. Σ, matching any alphabet
5. Σ^*, matching any finite string

Compound Patterns

1. $a \in \Sigma$, $L(a) = \{a\}$
2. ε, $L(\varepsilon) = \{\varepsilon\}$
3. $\varnothing$, $L(\varnothing) = \varnothing$
4. Σ, matching any alphabet
5. Σ^*, matching any finite string
6. x matches $\alpha + \beta$ if $L(\alpha + \beta) = L(\alpha) \cup L(\beta)$
7. x matches $\alpha \cap \beta$ if $L(\alpha \cap \beta) = L(\alpha) \cap L(\beta)$
8. x matches $\alpha\beta$ if $x = yz$ where $L(\alpha\beta) = L(\alpha)L(\beta)$
9. x matches $\overline{\alpha}$ if $L(\overline{\alpha}) = \overline{L(\alpha)} = \Sigma^* \setminus L(\alpha)$

Compound Patterns

1. $a \in \Sigma$, $L(a) = \{a\}$
2. ε, $L(\varepsilon) = \{\varepsilon\}$
3. $\varnothing$, $L(\varnothing) = \varnothing$
4. Σ, matching any alphabet
5. Σ^*, matching any finite string
6. x matches $\alpha + \beta$ if $L(\alpha + \beta) = L(\alpha) \cup L(\beta)$
7. x matches $\alpha \cap \beta$ if $L(\alpha \cap \beta) = L(\alpha) \cap L(\beta)$
8. x matches $\alpha\beta$ if $x = yz$ where $L(\alpha\beta) = L(\alpha)L(\beta)$
9. x matches $\overline{\alpha}$ if $L(\overline{\alpha}) = \overline{L(\alpha)} = \Sigma^* \setminus L(\alpha)$
10. x matches α^* if x can be expressed as zero or more of strings that match α, i.e., $L(\alpha^*) = L(\alpha)^*$
11. x matches α^+ if x can be expressed as one or more of strings that match α, i.e., $L(\alpha^+) = L(\alpha)^+$

Examples of Patterns

Given patterns for the following languages

Examples of Patterns

Given patterns for the following languages

- All words containing at least one a.

Examples of Patterns

Given patterns for the following languages

- All words containing at least one a. $\Sigma^* a \Sigma^*$

Examples of Patterns

Given patterns for the following languages

- All words containing at least one a. $\Sigma^* a \Sigma^*$
- Words containing at least one a or one b.
- Words containing at least one a and one b.

Examples of Patterns

Given patterns for the following languages

- All words containing at least one a. $\Sigma^* a \Sigma^*$
- Words containing at least one a or one b.
- Words containing at least one a and one b.
- Words of even length.

Examples of Patterns

Given patterns for the following languages

- All words containing at least one a. $\Sigma^* a \Sigma^*$
- Words containing at least one a or one b.
- Words containing at least one a and one b.
- Words of even length. $(\Sigma\Sigma)^*$
- Strings containing the substring $abaab$?
- Strings containing no occurrence of a

Examples of Patterns

Given patterns for the following languages

- All words containing at least one a. $\Sigma^* a \Sigma^*$
- Words containing at least one a or one b.
- Words containing at least one a and one b.
- Words of even length. $(\Sigma\Sigma)^*$
- Strings containing the substring $abaab$?
- Strings containing no occurrence of a $(\Sigma \cap \overline{a})^*$
- Strings where every occurence of a is followed by an occurence of b.

Examples of Patterns

Given patterns for the following languages

- All words containing at least one a. $\Sigma^* a \Sigma^*$
- Words containing at least one a or one b.
- Words containing at least one a and one b.
- Words of even length. $(\Sigma\Sigma)^*$
- Strings containing the substring $abaab$?
- Strings containing no occurrence of a $(\Sigma \cap \overline{a})^*$
- Strings where every occurence of a is followed by an occurence of b.

Express each of these languages as a pattern.

Questions

- How hard is to determine whether a given string x matches a pattern?
- Is every set of strings represented by some pattern?

Questions

- How hard is to determine whether a given string x matches a pattern?
- Is every set of strings represented by some pattern?
- Patterns $\alpha \equiv \beta$ if $L(\alpha) = L(\beta)$. Can you tell whether two given patterns are equivalent?

Questions

- How hard is to determine whether a given string x matches a pattern?
- Is every set of strings represented by some pattern?
- Patterns $\alpha \equiv \beta$ if $L(\alpha) = L(\beta)$. Can you tell whether two given patterns are equivalent?
- Which of the operators are redundant?

Questions

- How hard is to determine whether a given string x matches a pattern?
- Is every set of strings represented by some pattern?
- Patterns $\alpha \equiv \beta$ if $L(\alpha) = L(\beta)$. Can you tell whether two given patterns are equivalent?
- Which of the operators are redundant?
  - ε

Questions

- How hard is to determine whether a given string x matches a pattern?
- Is every set of strings represented by some pattern?
- Patterns $\alpha \equiv \beta$ if $L(\alpha) = L(\beta)$. Can you tell whether two given patterns are equivalent?
- Which of the operators are redundant?
  - $\varepsilon = \overline{\Sigma\Sigma^*}$

Questions

- How hard is to determine whether a given string x matches a pattern?
- Is every set of strings represented by some pattern?
- Patterns $\alpha \equiv \beta$ if $L(\alpha) = L(\beta)$. Can you tell whether two given patterns are equivalent?
- Which of the operators are redundant?
  - $\varepsilon = \overline{\Sigma\Sigma^*}$
  - Σ^*

Questions

- How hard is to determine whether a given string x matches a pattern?
- Is every set of strings represented by some pattern?
- Patterns $\alpha \equiv \beta$ if $L(\alpha) = L(\beta)$. Can you tell whether two given patterns are equivalent?
- Which of the operators are redundant?
  - $\varepsilon = \overline{\Sigma\Sigma^*}$
  - $\Sigma^* = (\Sigma)^*$

Questions

- How hard is to determine whether a given string x matches a pattern?
- Is every set of strings represented by some pattern?
- Patterns $\alpha \equiv \beta$ if $L(\alpha) = L(\beta)$. Can you tell whether two given patterns are equivalent?
- Which of the operators are redundant?
  - $\varepsilon = \overline{\Sigma\Sigma^*}$
  - $\Sigma^* = (\Sigma)^*$
  - Σ

Questions

- How hard is to determine whether a given string x matches a pattern?
- Is every set of strings represented by some pattern?
- Patterns $\alpha \equiv \beta$ if $L(\alpha) = L(\beta)$. Can you tell whether two given patterns are equivalent?
- Which of the operators are redundant?
  - $\varepsilon = \overline{\Sigma\Sigma^*}$
  - $\Sigma^* = (\Sigma)^*$
  - $\Sigma = a_1 + \cdots + a_k$

Questions

- How hard is to determine whether a given string x matches a pattern?
- Is every set of strings represented by some pattern?
- Patterns $\alpha \equiv \beta$ if $L(\alpha) = L(\beta)$. Can you tell whether two given patterns are equivalent?
- Which of the operators are redundant?
  - $\varepsilon = \overline{\Sigma\Sigma^*}$
  - $\Sigma^* = (\Sigma)^*$
  - $\Sigma = a_1 + \cdots + a_k$
  - $\alpha \cap \beta$

Questions

- How hard is to determine whether a given string x matches a pattern?
- Is every set of strings represented by some pattern?
- Patterns $\alpha \equiv \beta$ if $L(\alpha) = L(\beta)$. Can you tell whether two given patterns are equivalent?
- Which of the operators are redundant?
  - $\varepsilon = \overline{\Sigma\Sigma^*}$
  - $\Sigma^* = (\Sigma)^*$
  - $\Sigma = a_1 + \cdots + a_k$
  - $\alpha \cap \beta = \overline{\overline{\alpha} + \overline{\beta}}$

Questions

- How hard is to determine whether a given string x matches a pattern?
- Is every set of strings represented by some pattern?
- Patterns $\alpha \equiv \beta$ if $L(\alpha) = L(\beta)$. Can you tell whether two given patterns are equivalent?
- Which of the operators are redundant?
  - $\varepsilon = \overline{\Sigma\Sigma^*}$
  - $\Sigma^* = (\Sigma)^*$
  - $\Sigma = a_1 + \cdots + a_k$
  - $\alpha \cap \beta = \overline{\overline{\alpha} + \overline{\beta}}$
- Removing redundancy: Useful theoretically at the expense of losing succinctness.

Questions

- How hard is to determine whether a given string x matches a pattern?
- Is every set of strings represented by some pattern?
- Patterns $\alpha \equiv \beta$ if $L(\alpha) = L(\beta)$. Can you tell whether two given patterns are equivalent?
- Which of the operators are redundant?
  - $\varepsilon = \overline{\Sigma\Sigma^*}$
  - $\Sigma^* = (\Sigma)^*$
  - $\Sigma = a_1 + \cdots + a_k$
  - $\alpha \cap \beta = \overline{\overline{\alpha} + \overline{\beta}}$
- Removing redundancy: Useful theoretically at the expense of losing succinctness.
- Can you get rid of complementation?

Regular expressions

For a regular expression E we write $L(E)$ for its language. The set of valid regular expressions RegEx can be defined recursively as the following:

	Syntax	Semantics
Empty String	ϵ	$L(\epsilon) = \{\epsilon\}$
Empty Set	$\varnothing$	$L(\varnothing) = \varnothing$
Single Letter	a	$L(a) = \{a\}$
Union	$E + F$	$L(E + F) = L(E) \cup L(F)$
Concatenation	$E.F$	$L(E.F) = L(E) \circ L(F)$
Kleene Star	E^*	$L(E)^*$

Regular expressions

For a regular expression E we write $L(E)$ for its language. The set of valid regular expressions RegEx can be defined recursively as the following:

	Syntax	Semantics
Empty String	ϵ	$L(\epsilon) = \{\epsilon\}$
Empty Set	$\varnothing$	$L(\varnothing) = \varnothing$
Single Letter	a	$L(a) = \{a\}$
Union	$E + F$	$L(E + F) = L(E) \cup L(F)$
Concatenation	$E.F$	$L(E.F) = L(E) \circ L(F)$
Kleene Star	E^*	$L(E)^*$

Associativity of $+$ and $\circ$:

- $L(\alpha + (\beta + \gamma)) = L((\alpha + \beta) + \gamma)$
- $L(\alpha(\beta\gamma)) = L((\alpha\beta)\gamma)$

Regular expressions

For a regular expression E we write $L(E)$ for its language. The set of valid regular expressions RegEx can be defined recursively as the following:

	Syntax	Semantics
Empty String	ϵ	$L(\epsilon) = \{\epsilon\}$
Empty Set	$\varnothing$	$L(\varnothing) = \varnothing$
Single Letter	a	$L(a) = \{a\}$
Union	$E + F$	$L(E + F) = L(E) \cup L(F)$
Concatenation	$E.F$	$L(E.F) = L(E) \circ L(F)$
Kleene Star	E^*	$L(E)^*$

Associativity of $+$ and $\circ$:

- $L(\alpha + (\beta + \gamma)) = L((\alpha + \beta) + \gamma)$
- $L(\alpha(\beta\gamma)) = L((\alpha\beta)\gamma)$
- $\alpha + (\beta\gamma) \neq (\alpha + \beta)\gamma,$

Regular expressions

For a regular expression E we write $L(E)$ for its language. The set of valid regular expressions RegEx can be defined recursively as the following:

	Syntax	Semantics
Empty String	ϵ	$L(\epsilon) = \{\epsilon\}$
Empty Set	$\varnothing$	$L(\varnothing) = \varnothing$
Single Letter	a	$L(a) = \{a\}$
Union	$E + F$	$L(E + F) = L(E) \cup L(F)$
Concatenation	$E.F$	$L(E.F) = L(E) \circ L(F)$
Kleene Star	E^*	$L(E)^*$

Associativity of $+$ and $\circ$:

- $L(\alpha + (\beta + \gamma)) = L((\alpha + \beta) + \gamma)$
- $L(\alpha(\beta\gamma)) = L((\alpha\beta)\gamma)$
- $\alpha + (\beta\gamma) \neq (\alpha + \beta)\gamma$, $\alpha + \beta^* \neq (\alpha + \beta)^*$

Regular expressions

For a regular expression E we write $L(E)$ for its language. The set of valid regular expressions RegEx can be defined recursively as the following:

	Syntax	Semantics
Empty String	ϵ	$L(\epsilon) = \{\epsilon\}$
Empty Set	$\varnothing$	$L(\varnothing) = \varnothing$
Single Letter	a	$L(a) = \{a\}$
Union	$E + F$	$L(E + F) = L(E) \cup L(F)$
Concatenation	$E.F$	$L(E.F) = L(E) \circ L(F)$
Kleene Star	E^*	$L(E)^*$

Associativity of $+$ and $\circ$:

- $L(\alpha + (\beta + \gamma)) = L((\alpha + \beta) + \gamma)$
- $L(\alpha(\beta\gamma)) = L((\alpha\beta)\gamma)$
- $\alpha + (\beta\gamma) \neq (\alpha + \beta)\gamma$, $\alpha + \beta^* \neq (\alpha + \beta)^*$

Precedence rules: $* > \circ > +$

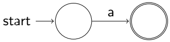
Language defined by regular expression

Lemma

The language defined by any regular expression is regular.

Example

$$(a + b)^*$$



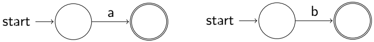
Language defined by regular expression

Lemma

The language defined by any regular expression is regular.

Example

$$(a+b)^*$$



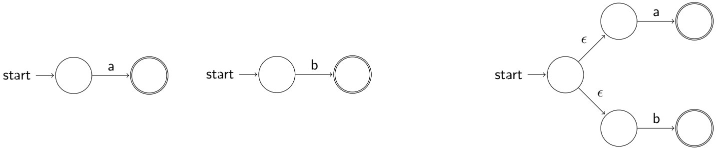
Language defined by regular expression

Lemma

The language defined by any regular expression is regular.

Example

$$(a + b)^*$$



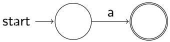
Language defined by regular expression

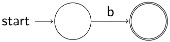
Lemma

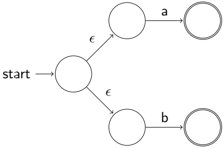
The language defined by any regular expression is regular.

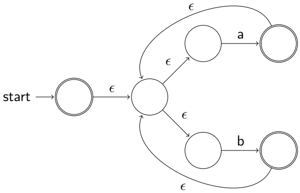
Example

$$(a+b)^*$$









Language defined by regular expression

Lemma

The language defined by any regular expression is regular.

Language defined by regular expression

Lemma

The language defined by any regular expression is regular.

Proof idea

Language defined by regular expression

Lemma

The language defined by any regular expression is regular.

Proof idea

It is easy to construct NFAs for 1.,2.,3.

Language defined by regular expression

Lemma

The language defined by any regular expression is regular.

Proof idea

It is easy to construct NFAs for 1.,2.,3.

If we inductively have NFAs for $L(R_1), L(R_2)$ then we can create an NFA for $L(R_1 + R_2)$ and $L(R_1 \circ R_2)$.

Language defined by regular expression

Lemma

The language defined by any regular expression is regular.

Proof idea

It is easy to construct NFAs for 1.,2.,3.

If we inductively have NFAs for $L(R_1), L(R_2)$ then we can create an NFA for $L(R_1 + R_2)$ and $L(R_1 \circ R_2)$.

Similarly, if we inductively have NFAs for $L(R_1)$ then we can create an NFA for $(L(R_1))^*$

Language defined by regular expression

Lemma

The language defined by any regular expression is regular.

Proof idea

It is easy to construct NFAs for 1.,2.,3.

If we inductively have NFAs for $L(R_1), L(R_2)$ then we can create an NFA for $L(R_1 + R_2)$ and $L(R_1 \circ R_2)$.

Similarly, if we inductively have NFAs for $L(R_1)$ then we can create an NFA for $(L(R_1))^*$

What about the converse?

Example

- $(aaa)^* + (aaaaa)^*$
- $(11+0)^*(00+1)^*$

A few axioms we can use for simplification

Associativity

$$\alpha + (\beta + \gamma) \equiv (\alpha + \beta) + \gamma$$
$$\alpha(\beta\gamma) \equiv (\alpha\beta)\gamma$$

A few axioms we can use for simplification

Associativity	$\alpha + (\beta + \gamma) \equiv (\alpha + \beta) + \gamma$
	$\alpha(\beta\gamma) \equiv (\alpha\beta)\gamma$
Commutativity	$\alpha + \beta \equiv \beta + \alpha$
Identity	$\alpha + \varnothing \equiv \alpha$
Idempotent	$\alpha + \alpha \equiv \alpha$

A few axioms we can use for simplification

Associativity	$\alpha + (\beta + \gamma) \equiv (\alpha + \beta) + \gamma$
	$\alpha(\beta\gamma) \equiv (\alpha\beta)\gamma$
Commutativity	$\alpha + \beta \equiv \beta + \alpha$
Identity	$\alpha + \varnothing \equiv \alpha$
Idempotent	$\alpha + \alpha \equiv \alpha$
Left Distributivity	$\alpha(\beta + \gamma) \equiv \alpha\beta + \alpha\gamma$
Right Distributivity	$(\alpha + \beta)\gamma \equiv \alpha\gamma + \beta\gamma$

A few axioms we can use for simplification

Associativity	$\alpha + (\beta + \gamma) \equiv (\alpha + \beta) + \gamma$
	$\alpha(\beta\gamma) \equiv (\alpha\beta)\gamma$
Commutativity	$\alpha + \beta \equiv \beta + \alpha$
Identity	$\alpha + \varnothing \equiv \alpha$
Idempotent	$\alpha + \alpha \equiv \alpha$
Left Distributivity	$\alpha(\beta + \gamma) \equiv \alpha\beta + \alpha\gamma$
Right Distributivity	$(\alpha + \beta)\gamma \equiv \alpha\gamma + \beta\gamma$
Closure	$\epsilon + \alpha\alpha^* \equiv \alpha^*$; $\epsilon + \alpha^*\alpha \equiv \alpha^*$

A few axioms we can use for simplification

Associativity	$\alpha + (\beta + \gamma) \equiv (\alpha + \beta) + \gamma$
	$\alpha(\beta\gamma) \equiv (\alpha\beta)\gamma$
Commutativity	$\alpha + \beta \equiv \beta + \alpha$
Identity	$\alpha + \emptyset \equiv \alpha$
Idempotent	$\alpha + \alpha \equiv \alpha$
Left Distributivity	$\alpha(\beta + \gamma) \equiv \alpha\beta + \alpha\gamma$
Right Distributivity	$(\alpha + \beta)\gamma \equiv \alpha\gamma + \beta\gamma$
Closure	$\epsilon + \alpha\alpha^* \equiv \alpha^*;\ \epsilon + \alpha^*\alpha \equiv \alpha^*$
DeMorgan-type laws	$(\alpha + \beta)^* = (\alpha^*\beta^*)^*$

A few axioms we can use for simplification

Associativity	$\alpha + (\beta + \gamma) \equiv (\alpha + \beta) + \gamma$
	$\alpha(\beta\gamma) \equiv (\alpha\beta)\gamma$
Commutativity	$\alpha + \beta \equiv \beta + \alpha$
Identity	$\alpha + \varnothing \equiv \alpha$
Idempotent	$\alpha + \alpha \equiv \alpha$
Left Distributivity	$\alpha(\beta + \gamma) \equiv \alpha\beta + \alpha\gamma$
Right Distributivity	$(\alpha + \beta)\gamma \equiv \alpha\gamma + \beta\gamma$
Closure	$\epsilon + \alpha\alpha^* \equiv \alpha^*$; $\epsilon + \alpha^*\alpha \equiv \alpha^*$
DeMorgan-type laws	$(\alpha + \beta)^* = (\alpha^*\beta^*)^*$
Subset order	$\beta + \alpha\gamma \leq \gamma \implies \alpha^*\beta \leq \gamma$
	$\beta + \gamma\alpha \leq \gamma \implies \beta\alpha^* \leq \gamma$

A few axioms we can use for simplification

Associativity	$\alpha + (\beta + \gamma) \equiv (\alpha + \beta) + \gamma$
	$\alpha(\beta\gamma) \equiv (\alpha\beta)\gamma$
Commutativity	$\alpha + \beta \equiv \beta + \alpha$
Identity	$\alpha + \varnothing \equiv \alpha$
Idempotent	$\alpha + \alpha \equiv \alpha$
Left Distributivity	$\alpha(\beta + \gamma) \equiv \alpha\beta + \alpha\gamma$
Right Distributivity	$(\alpha + \beta)\gamma \equiv \alpha\gamma + \beta\gamma$
Closure	$\epsilon + \alpha\alpha^* \equiv \alpha^*$; $\epsilon + \alpha^*\alpha \equiv \alpha^*$
DeMorgan-type laws	$(\alpha + \beta)^* = (\alpha^*\beta^*)^*$
Subset order	$\beta + \alpha\gamma \le \gamma \implies \alpha^*\beta \le \gamma$
	$\beta + \gamma\alpha \le \gamma \implies \beta\alpha^* \le \gamma$

where

$$\begin{aligned} \alpha \le \beta &\iff L(\alpha) \subseteq L(\beta) \\ &\iff L(\alpha + \beta) = L(\beta) \\ &\iff \alpha + \beta = \beta \end{aligned}$$

A few consequences that follow

Exercise!

$$
\begin{aligned}
(\alpha\beta)^*\alpha &\equiv \alpha(\beta\alpha)^* \\
(\alpha^*\beta)^*\alpha^* &\equiv (\alpha+\beta)^* \\
\alpha^*(\beta\alpha^*)^* &\equiv (\alpha+\beta)^* \\
(\epsilon+\alpha)^* &\equiv \alpha^* \\
\alpha\alpha^* &\equiv \alpha^*\alpha
\end{aligned}
$$

Example

- $(aaa)^* + (aaaaa)^*$
- $(11 + 0)^*(00 + 1)^*$
- $(1 + 01 + 001)^*(\varepsilon + 0 + 00)$

Example

- $(aaa)^* + (aaaaa)^*$
- $(11+0)^*(00+1)^*$
- $(1+01+001)^*(\varepsilon+0+00)$

$$(1+01+001)^*(\varepsilon+0+00) \quad \equiv \quad ((\varepsilon+0+00)1)^*(\varepsilon+0+00)$$

Example

- $(aaa)^* + (aaaaa)^*$
- $(11 + 0)^*(00 + 1)^*$
- $(1 + 01 + 001)^*(\varepsilon + 0 + 00)$

$$\begin{aligned}(1 + 01 + 001)^*(\varepsilon + 0 + 00) &\equiv ((\varepsilon + 0 + 00)1)^*(\varepsilon + 0 + 00)\\ &\equiv ((\varepsilon + 0)(\varepsilon + 0)1)^*(\varepsilon + 0)(\varepsilon + 0)\end{aligned}$$

Example

- $(aaa)^* + (aaaaa)^*$
- $(11 + 0)^*(00 + 1)^*$
- $(1 + 01 + 001)^*(\varepsilon + 0 + 00)$

$$
\begin{aligned}
(1 + 01 + 001)^*(\varepsilon + 0 + 00) &\equiv ((\varepsilon + 0 + 00)1)^*(\varepsilon + 0 + 00) \\
&\equiv ((\varepsilon + 0)(\varepsilon + 0)1)^*(\varepsilon + 0)(\varepsilon + 0)
\end{aligned}
$$

$(1 + 01 + 001)^*(\varepsilon + 0 + 00)$ = all strings over $\{0, 1\}$ with no substring of more than two adjacent 0's.

DFA to regular expression

Lemma

Any regular language can be specified by a regular expression

DFA to regular expression

Lemma

Any regular language can be specified by a regular expression

Want: Given any DFA, convert it into a regular expression.

Lemma

Given any DFA A, we can obtain a regular expression, say R_A, such that $L(A) = L(R_A)$.

Computing with labelled graphs

Lemma

Any regular language can be specified by a regular expression

Want: Given any DFA, convert it into a regular expression.

Lemma

Given any DFA A, we can obtain a regular expression, say R_A, such that $L(A) = L(R_A)$.