Regular Languages and Expressions

Lecture 2
Thursday, August 27, 2020
2.1
Regular Languages
Regular Languages

A class of simple but useful languages.

The set of **regular languages** over some alphabet $\Sigma$ is defined inductively as:

1. $\emptyset$ is a regular language.
2. $\{\epsilon\}$ is a regular language.
3. $\{a\}$ is a regular language for each $a \in \Sigma$. Interpreting $a$ as string of length 1.
4. If $L_1, L_2$ are regular then $L_1 \cup L_2$ is regular.
5. If $L_1, L_2$ are regular then $L_1L_2$ is regular.
6. If $L$ is regular, then $L^* = \bigcup_{n \geq 0} L^n$ is regular.
   The $^*$ operator name is **Kleene star**.
7. If $L$ is regular, then so is $\overline{L} = \Sigma^* \setminus L$.

Regular languages are closed under operations of union, concatenation and Kleene star.
Regular Languages

A class of simple but useful languages.
The set of regular languages over some alphabet $\Sigma$ is defined inductively as:

1. $\emptyset$ is a regular language.
2. $\{\epsilon\}$ is a regular language.
3. $\{a\}$ is a regular language for each $a \in \Sigma$. Interpreting $a$ as string of length 1.
4. If $L_1, L_2$ are regular then $L_1 \cup L_2$ is regular.
5. If $L_1, L_2$ are regular then $L_1L_2$ is regular.
6. If $L$ is regular, then $L^* = \bigcup_{n \geq 0} L^n$ is regular.
   The $\cdot^*$ operator name is Kleene star.
7. If $L$ is regular, then so is $\overline{L} = \Sigma^* \setminus L$.

Regular languages are closed under operations of union, concatenation and Kleene star.
Regular Languages

A class of simple but useful languages.
The set of regular languages over some alphabet $\Sigma$ is defined inductively as:

1. $\emptyset$ is a regular language.
2. $\{\epsilon\}$ is a regular language.
3. $\{a\}$ is a regular language for each $a \in \Sigma$. Interpreting $a$ as string of length 1.
4. If $L_1, L_2$ are regular then $L_1 \cup L_2$ is regular.
5. If $L_1, L_2$ are regular then $L_1L_2$ is regular.
6. If $L$ is regular, then $L^* = \cup_{n \geq 0} L^n$ is regular.
   The $^*$ operator name is Kleene star.
7. If $L$ is regular, then so is $\overline{L} = \Sigma^* \setminus L$.

Regular languages are closed under operations of union, concatenation and Kleene star.
Regular Languages

A class of simple but useful languages.
The set of regular languages over some alphabet $\Sigma$ is defined inductively as:

1. $\emptyset$ is a regular language.
2. $\{\epsilon\}$ is a regular language.
3. $\{a\}$ is a regular language for each $a \in \Sigma$. Interpreting $a$ as string of length 1.
4. If $L_1, L_2$ are regular then $L_1 \cup L_2$ is regular.
5. If $L_1, L_2$ are regular then $L_1L_2$ is regular.
6. If $L$ is regular, then $L^* = \bigcup_{n \geq 0} L^n$ is regular.
   The $\cdot^*$ operator name is **Kleene star**.
7. If $L$ is regular, then so is $\overline{L} = \Sigma^* \setminus L$.

Regular languages are closed under operations of union, concatenation and Kleene star.
Regular Languages

A class of simple but useful languages.

The set of regular languages over some alphabet $\Sigma$ is defined inductively as:

1. $\emptyset$ is a regular language.
2. $\{\epsilon\}$ is a regular language.
3. $\{a\}$ is a regular language for each $a \in \Sigma$. Interpreting $a$ as string of length 1.
4. If $L_1, L_2$ are regular then $L_1 \cup L_2$ is regular.
5. If $L_1, L_2$ are regular then $L_1L_2$ is regular.
6. If $L$ is regular, then $L^* = \bigcup_{n \geq 0} L^n$ is regular.
   The $\cdot^*$ operator name is Kleene star.
7. If $L$ is regular, then so is $\overline{L} = \Sigma^* \setminus L$.

Regular languages are closed under operations of union, concatenation and Kleene star.
Regular Languages

A class of simple but useful languages.
The set of regular languages over some alphabet $\Sigma$ is defined inductively as:

1. $\emptyset$ is a regular language.
2. $\{\varepsilon\}$ is a regular language.
3. $\{a\}$ is a regular language for each $a \in \Sigma$. Interpreting $a$ as string of length 1.
4. If $L_1, L_2$ are regular then $L_1 \cup L_2$ is regular.
5. If $L_1, L_2$ are regular then $L_1L_2$ is regular.
6. If $L$ is regular, then $L^* = \bigcup_{n \geq 0} L^n$ is regular.
   The $\cdot^*$ operator name is **Kleene star**.
7. If $L$ is regular, then so is $\overline{L} = \Sigma^* \setminus L$.

Regular languages are closed under operations of union, concatenation and Kleene star.
Regular Languages

Have basic operations to build regular languages.

**Important:** Any language generated by a finite sequence of such operations is regular.

**Lemma**

Let $L_1, L_2, \ldots$, be regular languages over alphabet $\Sigma$. Then the language $\bigcup_{i=1}^{\infty} L_i$ is not necessarily regular.
Some simple regular languages

Lemma
If \( w \) is a string then \( L = \{ w \} \) is regular.

Example: \{aba\} or \{abbabbab\}. Why?

Lemma
Every finite language \( L \) is regular.

Examples: \( L = \{a, abaab, aba\} \). \( L = \{w \mid |w| \leq 100\} \). Why?
Some simple regular languages

Lemma

If $w$ is a string then $L = \{w\}$ is regular.

Example: $\{aba\}$ or $\{abbabbab\}$. Why?

Lemma

Every finite language $L$ is regular.

Examples: $L = \{a, abaab, aba\}$. $L = \{w \mid |w| \leq 100\}$. Why?
More Examples

- $\{w \mid w$ is a keyword in Python program$\}$
- $\{w \mid w$ is a valid date of the form mm/dd/yy$\}$
- $\{w \mid w$ describes a valid Roman numeral$\}$
- $\{w \mid w$ contains ”CS374” as a substring$\}$. 
Review questions

1. $L_1 \subseteq \{0, 1\}^*$ be a finite language. $L_1$ is a set with finite number of strings. T/F?

2. $L_2 = \{0^i \mid i = 0, 1, \ldots, \infty\}$. The language $L_2$ is regular. T/F?

3. $L_3 = \{0^{2i} \mid i = 0, 1, \ldots, \infty\}$. The language $L_3$ is regular. T/F?

4. $L_4 = \{0^{17i} \mid i = 0, 1, \ldots, \infty\}$. The language $L_4$ is regular. T/F?

5. $L_5 = \{0^i \mid i \text{ is not divisible by } 17\}$. $L_5$ is regular. T/F?

6. $L_6 = \{0^i \mid i \text{ is divisible by } 2, 3, \text{ or } 5\}$. $L_6$ is regular. T/F?

7. $L_7 = \{0^i \mid i \text{ is divisible by } 2, 3, \text{ and } 5\}$. $L_7$ is regular. T/F?

8. $L_8 = \{0^i \mid i \text{ is divisible by } 2, 3, \text{ but not } 5\}$. $L_8$ is regular. T/F?

9. $L_9 = \{0^i1^i \mid i \text{ is divisible by } 2, 3, \text{ but not } 5\}$. $L_9$ is regular. T/F?

10. $L_{10} = \{w \in \{0, 1\}^* \mid w \text{ has at most } 374 \text{ 1s}\}$. $L_{10}$ is regular. T/F?
THE END

...(for now)
2.1.1
Regular Languages: Review questions
Review questions

1. Let $L_1 \subseteq \{0, 1\}^*$ be a finite language. $L_1$ is a set with finite number of strings. T/F?

2. Let $L_2 = \{0^i \mid i = 0, 1, \ldots, \infty\}$. The language $L_2$ is regular. T/F?

3. Let $L_3 = \{0^{2^i} \mid i = 0, 1, \ldots, \infty\}$. The language $L_3$ is regular. T/F?

4. Let $L_4 = \{0^{17^i} \mid i = 0, 1, \ldots, \infty\}$. The language $L_4$ is regular. T/F?

5. Let $L_5 = \{0^i \mid i \text{ is not divisible by 17}\}$. $L_5$ is regular. T/F?

6. Let $L_6 = \{0^i \mid i \text{ is divisible by 2, 3, or 5}\}$. $L_6$ is regular. T/F?

7. Let $L_7 = \{0^i \mid i \text{ is divisible by 2, 3, and 5}\}$. $L_7$ is regular. T/F?

8. Let $L_8 = \{0^i \mid i \text{ is divisible by 2, 3, but not 5}\}$. $L_8$ is regular. T/F?

9. Let $L_9 = \{0^{i1^i} \mid i \text{ is divisible by 2, 3, but not 5}\}$. $L_9$ is regular. T/F?

10. Let $L_{10} = \{w \in \{0, 1\}^* \mid w \text{ has at most 374 1s}\}$. $L_{10}$ is regular. T/F?
Review questions

1. $L_1 \subseteq \{0, 1\}^*$ be a finite language. $L_1$ is a set with finite number of strings. T/F?

2. $L_2 = \{0^i \mid i = 0, 1, \ldots, \infty\}$. The language $L_2$ is regular. T/F?

3. $L_3 = \{0^{2i} \mid i = 0, 1, \ldots, \infty\}$. The language $L_3$ is regular. T/F?

4. $L_4 = \{0^{17i} \mid i = 0, 1, \ldots, \infty\}$. The language $L_4$ is regular. T/F?

5. $L_5 = \{0^i \mid i \text{ is not divisible by } 17\}$. $L_5$ is regular. T/F?

6. $L_6 = \{0^i \mid i \text{ is divisible by } 2, 3, \text{ or } 5\}$. $L_6$ is regular. T/F?

7. $L_7 = \{0^i \mid i \text{ is divisible by } 2, 3, \text{ and } 5\}$. $L_7$ is regular. T/F?

8. $L_8 = \{0^i \mid i \text{ is divisible by } 2, 3, \text{ but not } 5\}$. $L_8$ is regular. T/F?

9. $L_9 = \{0^i1^i \mid i \text{ is divisible by } 2, 3, \text{ but not } 5\}$. $L_9$ is regular. T/F?

10. $L_{10} = \{w \in \{0, 1\}^* \mid w \text{ has at most } 374 \text{ } 1s\}$. $L_{10}$ is regular. T/F?
Review questions

1. \( L_1 \subseteq \{0, 1\}^* \) be a finite language. \( L_1 \) is a set with finite number of strings. T/F?

2. \( L_2 = \{0^i \mid i = 0, 1, \ldots, \infty\} \). The language \( L_2 \) is regular. T/F?

3. \( L_3 = \{0^{2i} \mid i = 0, 1, \ldots, \infty\} \). The language \( L_3 \) is regular. T/F?

4. \( L_4 = \{0^{17i} \mid i = 0, 1, \ldots, \infty\} \). The language \( L_4 \) is regular. T/F?

5. \( L_5 = \{0^i \mid \text{i is not divisible by 17}\} \). \( L_5 \) is regular. T/F?

6. \( L_6 = \{0^i \mid \text{i is divisible by 2, 3, or 5}\} \). \( L_6 \) is regular. T/F?

7. \( L_7 = \{0^i \mid \text{i is divisible by 2, 3, and 5}\} \). \( L_7 \) is regular. T/F?

8. \( L_8 = \{0^i \mid \text{i is divisible by 2, 3, but not 5}\} \). \( L_8 \) is regular. T/F?

9. \( L_9 = \{0^{i1^i} \mid \text{i is divisible by 2, 3, but not 5}\} \). \( L_9 \) is regular. T/F?

10. \( L_{10} = \{w \in \{0, 1\}^* \mid w \text{ has at most 374 1s}\} \). \( L_{10} \) is regular. T/F?
Review questions

1. $L_1 \subseteq \{0, 1\}^*$ be a finite language. $L_1$ is a set with finite number of strings. T/F?

2. $L_2 = \{0^i \mid i = 0, 1, \ldots, \infty\}$. The language $L_2$ is regular. T/F?

3. $L_3 = \{0^{2i} \mid i = 0, 1, \ldots, \infty\}$. The language $L_3$ is regular. T/F?

4. $L_4 = \{0^{17i} \mid i = 0, 1, \ldots, \infty\}$. The language $L_4$ is regular. T/F?

5. $L_5 = \{0^i \mid i \text{ is not divisible by 17}\}$. $L_5$ is regular. T/F?

6. $L_6 = \{0^i \mid i \text{ is divisible by 2, 3, or 5}\}$. $L_6$ is regular. T/F?

7. $L_7 = \{0^i \mid i \text{ is divisible by 2, 3, and 5}\}$. $L_7$ is regular. T/F?

8. $L_8 = \{0^i \mid i \text{ is divisible by 2, 3, but not 5}\}$. $L_8$ is regular. T/F?

9. $L_9 = \{0^i1^i \mid i \text{ is divisible by 2, 3, but not 5}\}$. $L_9$ is regular. T/F?

10. $L_{10} = \{w \in \{0, 1\}^* \mid w \text{ has at most 374 1s}\}$. $L_{10}$ is regular. T/F?
Review questions

1. $L_1 \subseteq \{0, 1\}^*$ be a finite language. $L_1$ is a set with finite number of strings. T/F?

2. $L_2 = \{0^i \mid i = 0, 1, \ldots, \infty\}$. The language $L_2$ is regular. T/F?

3. $L_3 = \{0^{2i} \mid i = 0, 1, \ldots, \infty\}$. The language $L_3$ is regular. T/F?

4. $L_4 = \{0^{17i} \mid i = 0, 1, \ldots, \infty\}$. The language $L_4$ is regular. T/F?

5. $L_5 = \{0^i \mid i \text{ is not divisible by 17}\}$. $L_5$ is regular. T/F?

6. $L_6 = \{0^i \mid i \text{ is divisible by 2, 3, or 5}\}$. $L_6$ is regular. T/F?

7. $L_7 = \{0^i \mid i \text{ is divisible by 2, 3, and 5}\}$. $L_7$ is regular. T/F?

8. $L_8 = \{0^i \mid i \text{ is divisible by 2, 3, but not 5}\}$. $L_8$ is regular. T/F?

9. $L_9 = \{0^{i1^i} \mid i \text{ is divisible by 2, 3, but not 5}\}$. $L_9$ is regular. T/F?

10. $L_{10} = \{w \in \{0, 1\}^* \mid w \text{ has at most 374 1s}\}$. $L_{10}$ is regular. T/F?
Review questions

1. \( L_1 \subseteq \{0, 1\}^\ast \) be a finite language. \( L_1 \) is a set with finite number of strings. T/F?

2. \( L_2 = \{0^i \mid i = 0, 1, \ldots, \infty\} \). The language \( L_2 \) is regular. T/F?

3. \( L_3 = \{0^{2i} \mid i = 0, 1, \ldots, \infty\} \). The language \( L_3 \) is regular. T/F?

4. \( L_4 = \{0^{17i} \mid i = 0, 1, \ldots, \infty\} \). The language \( L_4 \) is regular. T/F?

5. \( L_5 = \{0^i \mid i \text{ is not divisible by 17}\} \). \( L_5 \) is regular. T/F?

6. \( L_6 = \{0^i \mid i \text{ is divisible by 2, 3, or 5}\} \). \( L_6 \) is regular. T/F?

7. \( L_7 = \{0^i \mid i \text{ is divisible by 2, 3, and 5}\} \). \( L_7 \) is regular. T/F?

8. \( L_8 = \{0^i \mid i \text{ is divisible by 2, 3, but not 5}\} \). \( L_8 \) is regular. T/F?

9. \( L_9 = \{0^i1^i \mid i \text{ is divisible by 2, 3, but not 5}\} \). \( L_9 \) is regular. T/F?

10. \( L_{10} = \{w \in \{0, 1\}^\ast \mid w \text{ has at most 374 1s}\} \). \( L_{10} \) is regular. T/F?
Review questions

1. $L_1 \subseteq \{0, 1\}^*$ be a finite language. $L_1$ is a set with finite number of strings. T/F?

2. $L_2 = \{0^i \mid i = 0, 1, \ldots, \infty\}$. The language $L_2$ is regular. T/F?

3. $L_3 = \{0^{2i} \mid i = 0, 1, \ldots, \infty\}$. The language $L_3$ is regular. T/F?

4. $L_4 = \{0^{17i} \mid i = 0, 1, \ldots, \infty\}$. The language $L_4$ is regular. T/F?

5. $L_5 = \{0^i \mid i \text{ is not divisible by 17}\}$. $L_5$ is regular. T/F?

6. $L_6 = \{0^i \mid i \text{ is divisible by 2, 3, or 5}\}$. $L_6$ is regular. T/F?

7. $L_7 = \{0^i \mid i \text{ is divisible by 2, 3, and 5}\}$. $L_7$ is regular. T/F?

8. $L_8 = \{0^i \mid i \text{ is divisible by 2, 3, but not 5}\}$. $L_8$ is regular. T/F?

9. $L_9 = \{0^i1^i \mid i \text{ is divisible by 2, 3, but not 5}\}$. $L_9$ is regular. T/F?

10. $L_{10} = \{w \in \{0, 1\}^* \mid w \text{ has at most 374 1s}\}$. $L_{10}$ is regular. T/F?
Review questions

1. \( L_1 \subseteq \{0, 1\}^* \) be a finite language. \( L_1 \) is a set with finite number of strings. T/F?

2. \( L_2 = \{0^i \mid i = 0, 1, \ldots, \infty\} \). The language \( L_2 \) is regular. T/F?

3. \( L_3 = \{0^{2i} \mid i = 0, 1, \ldots, \infty\} \). The language \( L_3 \) is regular. T/F?

4. \( L_4 = \{0^{17i} \mid i = 0, 1, \ldots, \infty\} \). The language \( L_4 \) is regular. T/F?

5. \( L_5 = \{0^i \mid i \text{ is not divisible by 17}\} \). \( L_5 \) is regular. T/F?

6. \( L_6 = \{0^i \mid i \text{ is divisible by 2, 3, or 5}\} \). \( L_6 \) is regular. T/F?

7. \( L_7 = \{0^i \mid i \text{ is divisible by 2, 3, and 5}\} \). \( L_7 \) is regular. T/F?

8. \( L_8 = \{0^i \mid i \text{ is divisible by 2, 3, but not 5}\} \). \( L_8 \) is regular. T/F?

9. \( L_9 = \{0^i1^i \mid i \text{ is divisible by 2, 3, but not 5}\} \). \( L_9 \) is regular. T/F?

10. \( L_{10} = \{w \in \{0, 1\}^* \mid w \text{ has at most 374 1s}\} \). \( L_{10} \) is regular. T/F?
Review questions

1. \( L_1 \subseteq \{0, 1\}^* \) be a finite language. \( L_1 \) is a set with finite number of strings. T/F?

2. \( L_2 = \{0^i \mid i = 0, 1, \ldots, \infty\} \). The language \( L_2 \) is regular. T/F?

3. \( L_3 = \{0^{2i} \mid i = 0, 1, \ldots, \infty\} \). The language \( L_3 \) is regular. T/F?

4. \( L_4 = \{0^{17i} \mid i = 0, 1, \ldots, \infty\} \). The language \( L_4 \) is regular. T/F?

5. \( L_5 = \{0^i \mid i \text{ is not divisible by 17}\} \). \( L_5 \) is regular. T/F?

6. \( L_6 = \{0^i \mid i \text{ is divisible by 2, 3, or 5}\} \). \( L_6 \) is regular. T/F?

7. \( L_7 = \{0^i \mid i \text{ is divisible by 2, 3, and 5}\} \). \( L_7 \) is regular. T/F?

8. \( L_8 = \{0^i \mid i \text{ is divisible by 2, 3, but not 5}\} \). \( L_8 \) is regular. T/F?

9. \( L_9 = \{0^i1^i \mid i \text{ is divisible by 2, 3, but not 5}\} \). \( L_9 \) is regular. T/F?

10. \( L_{10} = \{w \in \{0, 1\}^* \mid w \text{ has at most } 374 \text{ 1s}\} \). \( L_{10} \) is regular. T/F?
Review questions

1. $L_1 \subseteq \{0, 1\}^*$ be a finite language. $L_1$ is a set with finite number of strings. T/F?

2. $L_2 = \{0^i \mid i = 0, 1, \ldots, \infty\}$. The language $L_2$ is regular. T/F?

3. $L_3 = \{0^{2i} \mid i = 0, 1, \ldots, \infty\}$. The language $L_3$ is regular. T/F?

4. $L_4 = \{0^{17i} \mid i = 0, 1, \ldots, \infty\}$. The language $L_4$ is regular. T/F?

5. $L_5 = \{0^i \mid i \text{ is not divisible by 17}\}$. $L_5$ is regular. T/F?

6. $L_6 = \{0^i \mid i \text{ is divisible by 2, 3, or 5}\}$. $L_6$ is regular. T/F?

7. $L_7 = \{0^i \mid i \text{ is divisible by 2, 3, and 5}\}$. $L_7$ is regular. T/F?

8. $L_8 = \{0^i \mid i \text{ is divisible by 2, 3, but not 5}\}$. $L_8$ is regular. T/F?

9. $L_9 = \{0^i1^i \mid i \text{ is divisible by 2, 3, but not 5}\}$. $L_9$ is regular. T/F?

10. $L_{10} = \{w \in \{0, 1\}^* \mid w \text{ has at most 374 1s}\}$. $L_{10}$ is regular. T/F?
THE END

...(for now)
2.2

Regular Expressions
Regular Expressions

A way to denote regular languages

- simple **patterns** to describe related strings
- useful in
  - text search (editors, Unix/grep, emacs)
  - compilers: lexical analysis
  - compact way to represent interesting/useful languages
  - dates back to 50’s: Stephen Kleene
    who has a star names after him.
Inductive Definition

A regular expression \( r \) over an alphabet \( \Sigma \) is one of the following:

**Base cases:**
- \( \emptyset \) denotes the language \( \emptyset \)
- \( \epsilon \) denotes the language \( \{\epsilon\} \).
- \( a \) denote the language \( \{a\} \).

**Inductive cases:** If \( r_1 \) and \( r_2 \) are regular expressions denoting languages \( R_1 \) and \( R_2 \) respectively then,
- \( (r_1 + r_2) \) denotes the language \( R_1 \cup R_2 \)
- \( (r_1 \cdot r_2) = r_1 \cdot r_2 = (r_1 r_2) \) denotes the language \( R_1 R_2 \)
- \( (r_1)^* \) denotes the language \( R_1^* \)
Inductive Definition

A regular expression $r$ over an alphabet $\Sigma$ is one of the following:

**Base cases:**
- $\emptyset$ denotes the language $\emptyset$
- $\epsilon$ denotes the language $\{\epsilon\}$.
- $a$ denote the language $\{a\}$.

**Inductive cases:** If $r_1$ and $r_2$ are regular expressions denoting languages $R_1$ and $R_2$ respectively then,
- $(r_1 + r_2)$ denotes the language $R_1 \cup R_2$
- $(r_1 \cdot r_2) = r_1 \cdot r_2 = (r_1 r_2)$ denotes the language $R_1 R_2$
- $(r_1)^*$ denotes the language $R_1^*$
Regular Languages vs Regular Expressions

<table>
<thead>
<tr>
<th>Regular Languages</th>
<th>Regular Expressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$ regular</td>
<td>$\emptyset$ denotes $\emptyset$</td>
</tr>
<tr>
<td>${\epsilon}$ regular</td>
<td>$\epsilon$ denotes ${\epsilon}$</td>
</tr>
<tr>
<td>${a}$ regular for $a \in \Sigma$</td>
<td>$a$ denote ${a}$</td>
</tr>
<tr>
<td>$R_1 \cup R_2$ regular if both are</td>
<td>$r_1 + r_2$ denotes $R_1 \cup R_2$</td>
</tr>
<tr>
<td>$R_1 R_2$ regular if both are</td>
<td>$r_1 \cdot r_2$ denotes $R_1 R_2$</td>
</tr>
<tr>
<td>$R^*$ is regular if $R$ is</td>
<td>$r^<em>$ denote $R^</em>$</td>
</tr>
</tbody>
</table>

Regular expressions denote regular languages — they explicitly show the operations that were used to form the language.
Notation and Parenthesis

- For a regular expression $r$, $L(r)$ is the language denoted by $r$. Multiple regular expressions can denote the same language!
  
  **Example:** $(0 + 1)$ and $(1 + 0)$ denote the same language $\{0, 1\}$

- Two regular expressions $r_1$ and $r_2$ are equivalent if $L(r_1) = L(r_2)$.

- Omit parenthesis by adopting precedence order: $\ast$, concatenate, $\ast$.
  
  **Example:** $r^\ast s + t = ((r^\ast)s) + t$

- Omit parenthesis by associativity of each of these operations.
  
  **Example:** $rst = (rs)t = r(st)$, $r + s + t = r + (s + t) = (r + s) + t$.

- Superscript $\ast$. For convenience, define $r^+ = rr^\ast$. Hence if $L(r) = R$ then $L(r^+) = R^+$.

- Other notation: $r + s, r \cup s, r | s$ all denote union. $rs$ is sometimes written as $r \cdot s$. 
For a regular expression \( r \), \( L(r) \) is the language denoted by \( r \). Multiple regular expressions can denote the same language!

**Example:** \((0 + 1)\) and \((1 + 0)\) denote same language \( \{0, 1\} \)

Two regular expressions \( r_1 \) and \( r_2 \) are **equivalent** if \( L(r_1) = L(r_2) \).

- Omit parenthesis by adopting precedence order: \( \ast \), concatenate, \(+\).
  
  **Example:** \( r^*s + t = ((r^*)s) + t \)

- Omit parenthesis by associativity of each of these operations.
  
  **Example:** \( rst = (rs)t = r(st) \), \( r + s + t = r + (s + t) = (r + s) + t \).

- Superscript \(+\). For convenience, define \( r^+ = rr^* \). Hence if \( L(r) = R \) then \( L(r^+) = R^+ \).

- Other notation: \( r + s \), \( r \cup s \), \( r | s \) all denote union. \( rs \) is sometimes written as \( r \circ s \).
For a regular expression $r$, $L(r)$ is the language denoted by $r$. Multiple regular expressions can denote the same language!

**Example:** $(0 + 1)$ and $(1 + 0)$ denote same language $\{0, 1\}$

Two regular expressions $r_1$ and $r_2$ are equivalent if $L(r_1) = L(r_2)$.

Omit parenthesis by adopting precedence order: $\ast$, concatenate, $+$. **Example:** $r^*s + t = ((r^*)s) + t$

Omit parenthesis by associativity of each of these operations. **Example:** $rst = (rs)t = r(st)$, $r + s + t = r + (s + t) = (r + s) + t$.

Superscript $\ast$. For convenience, define $r^+ = rr^*$. Hence if $L(r) = R$ then $L(r^+) = R^+$.

Other notation: $r + s$, $r \cup s$, $r|s$ all denote union. $rs$ is sometimes written as $r \cdot s$. 
For a regular expression $r$, $L(r)$ is the language denoted by $r$. Multiple regular expressions can denote the same language!

**Example:** $(0 + 1)$ and $(1 + 0)$ denote same language $\{0, 1\}$

Two regular expressions $r_1$ and $r_2$ are equivalent if $L(r_1) = L(r_2)$.

Omit parenthesis by adopting precedence order: $\ast$, concatenate, $\cup$.

**Example:** $r^*s + t = ((r^*)s) + t$

Omit parenthesis by associativity of each of these operations.

**Example:** $rst = (rs)t = r(st)$, $r + s + t = r + (s + t) = (r + s) + t$.

Superscript $\ast$. For convenience, define $r^+ = rr^*$. Hence if $L(r) = R$ then $L(r^+) = R^+$.

Other notation: $r + s$, $r \cup s$, $r | s$ all denote union. $rs$ is sometimes written as $r \cdot s$. 
Notation and Parenthesis

- For a regular expression $r$, $L(r)$ is the language denoted by $r$. Multiple regular expressions can denote the same language!
  - Example: $(0 + 1)$ and $(1 + 0)$ denote same language $\{0, 1\}$
- Two regular expressions $r_1$ and $r_2$ are equivalent if $L(r_1) = L(r_2)$.
- Omit parenthesis by adopting precedence order: $\ast$, concatenate, $+$.
  - Example: $r^*s + t = ((r^*)s) + t$
- Omit parenthesis by associativity of each of these operations.
  - Example: $rst = (rs)t = r(st)$, $r + s + t = r + (s + t) = (r + s) + t$.
- Superscript $+$. For convenience, define $r^+ = rr^*$. Hence if $L(r) = R$ then $L(r^+) = R^+$.
- Other notation: $r + s$, $r \cup s$, $r|s$ all denote union. $rs$ is sometimes written as $r \cdot s$. 
Notation and Parenthesis

- For a regular expression $r$, $L(r)$ is the language denoted by $r$. Multiple regular expressions can denote the same language!
  **Example:** $(0 + 1)$ and $(1 + 0)$ denote same language $\{0, 1\}$

- Two regular expressions $r_1$ and $r_2$ are equivalent if $L(r_1) = L(r_2)$.

- Omit parenthesis by adopting precedence order: $\ast$, concatenate, $\ast$.
  **Example:** $r \ast s + t = ((r \ast) s) + t$

- Omit parenthesis by associativity of each of these operations.
  **Example:** $rst = (rs) t = r(s t)$, $r + s + t = r + (s + t) = (r + s) + t$.

- Superscript $\ast$. For convenience, define $r^\ast = rr^\ast$. Hence if $L(r) = R$ then $L(r^\ast) = R^\ast$.

- Other notation: $r + s$, $r \cup s$, $r | s$ all denote union. $rs$ is sometimes written as $r \cdot s$. 
Skills

- Given a language \( L \) “in mind” (say an English description) we would like to write a regular expression for \( L \) (if possible)
- Given a regular expression \( r \) we would like to “understand” \( L(r) \) (say by giving an English description)
Skills

- Given a language $L$ “in mind” (say an English description) we would like to write a regular expression for $L$ (if possible)
- Given a regular expression $r$ we would like to “understand” $L(r)$ (say by giving an English description)
THE END

...(for now)
2.2.1
Some examples of regular expressions
Understanding regular expressions

- \((0 + 1)^*\): set of all strings over \(\{0, 1\}\)
- \((0 + 1)^*001(0 + 1)^*\): strings with 001 as substring
- \(0^* + (0^*10^*10^*10^*)^*\): strings with number of 1's divisible by 3
- \(\emptyset\): \(
\{\}
\)
- \((\epsilon + 1)(01)^*(\epsilon + 0)\): alternating 0s and 1s. Alternatively, no two consecutive 0s and no two consecutive 1s
- \((\epsilon + 0)(1 + 10)^*\): strings without two consecutive 0s.
Understanding regular expressions

- $(0 + 1)^*$: set of all strings over $\{0, 1\}$
- $(0 + 1)^*001(0 + 1)^*$: strings with 001 as substring
- $0^* + (0^*10^*10^*10^*)^*$: strings with number of 1's divisible by 3
- $\emptyset$: {}
- $(\epsilon + 1)(01)^*(\epsilon + 0)$: alternating 0s and 1s. Alternatively, no two consecutive 0s and no two consecutive 1s
- $(\epsilon + 0)(1 + 10)^*$: strings without two consecutive 0s.
Understanding regular expressions

- $(0 + 1)^*$: set of all strings over $\{0, 1\}$
- $(0 + 1)^*001(0 + 1)^*$: strings with $001$ as substring
- $0^* + (0^*10^*10^*10^*)^*$: strings with number of 1's divisible by 3
- $\emptyset$: $\{}$
- $(\epsilon + 1)(01)^*(\epsilon + 0)$: alternating 0s and 1s. Alternatively, no two consecutive 0s and no two consecutive 1s
- $(\epsilon + 0)(1 + 10)^*$: strings without two consecutive 0s.
Understanding regular expressions

- \((0 + 1)^*\): set of all strings over \(\{0, 1\}\)
- \((0 + 1)^*001(0 + 1)^*\): strings with 001 as substring
- \(0^* + (0^*10*10^*10^*)^*\): strings with number of 1's divisible by 3
- \(\emptyset\): \(
\)
- \((\epsilon + 1)(01)^*(\epsilon + 0)\): alternating 0s and 1s. Alternatively, no two consecutive 0s and no two consecutive 1s
- \((\epsilon + 0)(1 + 10)^*\): strings without two consecutive 0s.
Understanding regular expressions

- \((0 + 1)^*\): set of all strings over \(\{0, 1\}\)
- \((0 + 1)^*001(0 + 1)^*\): strings with 001 as substring
- \(0^* + (0^*10^*10^*10^*)^*\): strings with number of 1's divisible by 3
- \(\emptyset\): \(\{\}\)
- \((\epsilon + 1)(01)^*(\epsilon + 0)\): alternating 0s and 1s. Alternatively, no two consecutive 0s and no two consecutive 1s
- \((\epsilon + 0)(1 + 10)^*\): strings without two consecutive 0s.
Understanding regular expressions

- \((0 + 1)^*\): set of all strings over \(\{0, 1\}\)
- \((0 + 1)^*001(0 + 1)^*\): strings with 001 as substring
- \(0^* + (0^*10^*10^*10^*)^*\): strings with number of 1’s divisible by 3
- \(\emptyset\): \(\{\}\)
- \((\epsilon + 1)(01)^*(\epsilon + 0)\): alternating 0s and 1s. Alternatively, no two consecutive 0s and no two consecutive 1s
- \((\epsilon + 0)(1 + 10)^*\): strings without two consecutive 0s.
Understanding regular expressions

- \((0 + 1)^*\): set of all strings over \(\{0, 1\}\)
- \((0 + 1)^*001(0 + 1)^*\): strings with 001 as substring
- \(0^* + (0^*10^*10^*10^*)^*\): strings with number of 1's divisible by 3
- \(\emptyset\): \(\{\}\)
- \((\epsilon + 1)(01)^*(\epsilon + 0)\): alternating 0s and 1s. Alternatively, no two consecutive 0s and no two consecutive 1s
- \((\epsilon + 0)(1 + 10)^*\): strings without two consecutive 0s.
Understanding regular expressions

- \((0 + 1)^*\): set of all strings over \(\{0, 1\}\)
- \((0 + 1)^*001(0 + 1)^*\): strings with 001 as substring
- \(0^* + (0*10*10^*10^*)^*\): strings with number of 1's divisible by 3
- \(\emptyset\): \(
\) strings without two consecutive 0s and no two consecutive 1s
- \((\epsilon + 1)(01)(\epsilon + 0)^*\): alternating 0s and 1s. Alternatively, no two consecutive 0s and no two consecutive 1s
- \((\epsilon + 0)(1 + 10)^*\): strings without two consecutive 0s.
Understanding regular expressions

- \((0 + 1)^*\): set of all strings over \(\{0, 1\}\)
- \((0 + 1)^*001(0 + 1)^*\): strings with 001 as substring
- \(0^* + (0^*10^*10^*10^*)^*\): strings with number of 1’s divisible by 3
- \(\emptyset0\): \(\{\}\)
- \((\epsilon + 1)(01)(\epsilon + 0)\): alternating 0s and 1s. Alternatively, no two consecutive 0s and no two consecutive 1s
- \((\epsilon + 0)(1 + 10)^*\): strings without two consecutive 0s.
Understanding regular expressions

- $(0 + 1)^*$: set of all strings over $\{0, 1\}$
- $(0 + 1)^*001(0 + 1)^*$: strings with 001 as substring
- $0^* + (0*10*10*10^*)^*$: strings with number of 1's divisible by 3
- $\emptyset$: $\{\}$
- $(\epsilon + 1)(01)^*(\epsilon + 0)$: alternating 0s and 1s. Alternatively, no two consecutive 0s and no two consecutive 1s
- $(\epsilon + 0)(1 + 10)^*$: strings without two consecutive 0s.
Understanding regular expressions

- \((0 + 1)^*\): set of all strings over \(\{0, 1\}\)
- \((0 + 1)^*001(0 + 1)^*\): strings with 001 as substring
- \(0^* + (0^*10^*10^*10^*)^*\): strings with number of 1's divisible by 3
- \(\emptyset\): \(\{\}\)
- \((\epsilon + 1)(01)^*(\epsilon + 0)\): alternating 0s and 1s. Alternatively, no two consecutive 0s and no two consecutive 1s
- \((\epsilon + 0)(1 + 10)^*\): strings without two consecutive 0s.
Creating regular expressions

- bitstrings with the pattern 001 or the pattern 100 occurring as a substring
  one answer: $(0 + 1)^*001(0 + 1)^* + (0 + 1)^*100(0 + 1)^*$
- bitstrings with an even number of 1's
  one answer: $0^* + (0^*10^*10^*)^*$
- bitstrings with an odd number of 1’s
  one answer: $0^*1r$ where $r$ is solution to previous part
- bitstrings that do not contain 011 as a substring
- Hard: bitstrings with an odd number of 1s and an odd number of 0s.
Creating regular expressions

- bitstrings with the pattern 001 or the pattern 100 occurring as a substring
  one answer: $(0 + 1)^*001(0 + 1)^* + (0 + 1)^*100(0 + 1)^*$

- bitstrings with an even number of 1’s
  one answer: $0^* + (0*10*10^*)^*$

- bitstrings with an odd number of 1’s
  one answer: $0*1r$ where $r$ is solution to previous part

- bitstrings that do not contain 011 as a substring

- Hard: bitstrings with an odd number of 1s and an odd number of 0s.
Creating regular expressions

- bitstrings with the pattern 001 or the pattern 100 occurring as a substring
  one answer: \((0 + 1)^*001(0 + 1)^* + (0 + 1)^*100(0 + 1)^*\)

- bitstrings with an even number of 1’s
  one answer: \(0^* + (0^*10^*10^*)^*\)

- bitstrings with an odd number of 1’s
  one answer: \(0^*1r\) where \(r\) is solution to previous part

- bitstrings that do not contain 011 as a substring

- Hard: bitstrings with an odd number of 1s and an odd number of 0s.
Creating regular expressions

- bitstrings with the pattern 001 or the pattern 100 occurring as a substring
  one answer: \((0 + 1)^*001(0 + 1)^* + (0 + 1)^*100(0 + 1)^*\)

- bitstrings with an even number of 1’s
  one answer: \(0^* + (0^*10^*10^*)^*\)

- bitstrings with an odd number of 1’s
  one answer: \(0^*1r\) where \(r\) is solution to previous part

- bitstrings that do not contain 011 as a substring

- Hard: bitstrings with an odd number of 1s and an odd number of 0s.
Creating regular expressions

- bitstrings with the pattern 001 or the pattern 100 occurring as a substring
  one answer: \((0 + 1)^*001(0 + 1)^* + (0 + 1)^*100(0 + 1)^*\)

- bitstrings with an even number of 1’s
  one answer: \(0^* + (0^*10^*10^*)^*\)

- bitstrings with an odd number of 1’s
  one answer: \(0^*1r\) where \(r\) is solution to previous part

- bitstrings that do not contain 011 as a substring

- Hard: bitstrings with an odd number of 1s and an odd number of 0s.
Creating regular expressions

- bitstrings with the pattern 001 or the pattern 100 occurring as a substring
  one answer: \((0 + 1)^*001(0 + 1)^* + (0 + 1)^*100(0 + 1)^*\)

- bitstrings with an even number of 1’s
  one answer: \(0^* + (0^*10^*10^*)^*\)

- bitstrings with an odd number of 1’s
  one answer: \(0^*1r\) where \(r\) is solution to previous part

- bitstrings that do **not** contain 011 as a substring

- Hard: bitstrings with an odd number of 1s **and** an odd number of 0s.
Creating regular expressions

- bitstrings with the pattern 001 or the pattern 100 occurring as a substring
  one answer: $(0 + 1)^*001(0 + 1)^* + (0 + 1)^*100(0 + 1)^*$

- bitstrings with an even number of 1's
  one answer: $0^* + (0^*10^*10^*)^*$

- bitstrings with an odd number of 1's
  one answer: $0^*1r$ where $r$ is solution to previous part

- bitstrings that do not contain 011 as a substring

- Hard: bitstrings with an odd number of 1s and an odd number of 0s.
Creating regular expressions

- bitstrings with the pattern 001 or the pattern 100 occurring as a substring
  one answer: \((0 + 1)^*001(0 + 1)^* + (0 + 1)^*100(0 + 1)^*\)

- bitstrings with an even number of 1’s
  one answer: \(0^* + (0^*10^*10^*)^*\)

- bitstrings with an odd number of 1’s
  one answer: \(0^*1r\) where \(r\) is solution to previous part

- bitstrings that do not contain 011 as a substring

- Hard: bitstrings with an odd number of 1s and an odd number of 0s.
Bit strings with odd number of 0s and 1s

The regular expression is

\[(00 + 11)^*(01 + 10)\]
\[\left(00 + 11 + (01 + 10)(00 + 11)^*(01 + 10)\right)^*\]

(Solved using techniques to be presented in the following lectures...)
Regular expression identities

- $r^* r^* = r^*$ meaning for any regular expression $r$, $L(r^* r^*) = L(r^*)$
- $(r^*)^* = r^*$
- $rr^* = r^* r$
- $(rs)^* r = r(sr)^*$
- $(r + s)^* = (r^* s^*)^* = (r^* + s^*)^* = (r + s^*)^* = \ldots$

**Question**: How does one prove an identity?
By induction. On what? Length of $r$ since $r$ is a string obtained from specific inductive rules.
Regular expression identities

- $r^* r^* = r^*$ meaning for any regular expression $r$, $L(r^* r^*) = L(r^*)$
- $(r^*)^* = r^*$
- $rr^* = r^* r$
- $(rs)^* r = r(sr)^*$
- $(r + s)^* = (r^* s^*)^* = (r^* + s^*)^* = (r + s^*)^* = \ldots$

**Question:** How does one prove an identity?

By induction. On what? Length of $r$ since $r$ is a string obtained from specific inductive rules.
Regular expression identities

- $r^* r^* = r^*$ meaning for any regular expression $r$, $L(r^* r^*) = L(r^*)$
- $(r^*)^* = r^*$
- $rr^* = r^* r$
- $(rs)^* r = r(sr)^*$
- $(r + s)^* = (r^* s^*)^* = (r^* + s^*)^* = (r + s)^* = \ldots$

**Question:** How does one prove an identity?  
By induction. On what? Length of $r$ since $r$ is a string obtained from specific inductive rules.
Regular expression identities

- $r^*r^* = r^*$ meaning for any regular expression $r$, $L(r^*r^*) = L(r^*)$
- $(r^*)^* = r^*$
- $rr^* = r^*r$
- $(rs)^*r = r(sr)^*$
- $(r + s)^* = (r^*s^*)^* = (r^* + s^*)^* = (r + s^*)^* = \ldots$

**Question:** How does one prove an identity?

By induction. On what? Length of $r$ since $r$ is a string obtained from specific inductive rules.
THE END

...(for now)
2.2.2
An example of a non-regular language
A non-regular language and other closure properties

Consider \( L = \{ 0^n1^n \mid n \geq 0 \} = \{ \epsilon, 01, 0011, 000111, \ldots \} \).

**Theorem**

\[ L = \{ 0^n1^n \mid n \geq 0 \} = \{ \epsilon, 01, 0011, 000111, \ldots \}. \]

The language \( L \) is **not** a regular language.

How do we prove it?

Other questions:

- Suppose \( R_1 \) is regular and \( R_2 \) is regular. Is \( R_1 \cap R_2 \) regular?
- Suppose \( R_1 \) is regular is \( \overline{R_1} \) (complement of \( R_1 \)) regular?
A non-regular language and other closure properties

Consider $L = \{0^n1^n \mid n \geq 0\} = \{\epsilon, 01, 0011, 000111, \ldots\}$.

**Theorem**

$L = \{0^n1^n \mid n \geq 0\} = \{\epsilon, 01, 0011, 000111, \ldots\}$.  
*The language $L$ is not a regular language.*

How do we prove it?

Other questions:
- Suppose $R_1$ is regular and $R_2$ is regular. Is $R_1 \cap R_2$ regular?
- Suppose $R_1$ is regular is $\overline{R_1}$ (complement of $R_1$) regular?
A non-regular language and other closure properties

Consider \( L = \{0^n1^n \mid n \geq 0\} = \{\epsilon, 01, 0011, 000111, \ldots\} \).

**Theorem**

\( L = \{0^n1^n \mid n \geq 0\} = \{\epsilon, 01, 0011, 000111, \ldots\} \).

*The language \( L \) is not a regular language.*

How do we prove it?

Other questions:

- Suppose \( R_1 \) is regular and \( R_2 \) is regular. Is \( R_1 \cap R_2 \) regular?
- Suppose \( R_1 \) is regular is \( \overline{R_1} \) (complement of \( R_1 \)) regular?
A non-regular language and other closure properties

Consider \( L = \{0^n1^n \mid n \geq 0\} = \{\epsilon, 01, 0011, 000111, \ldots\} \).

**Theorem**

\[ L = \{0^n1^n \mid n \geq 0\} = \{\epsilon, 01, 0011, 000111, \ldots\} \]

*The language \( L \) is not a regular language.*

How do we prove it?

Other questions:

- Suppose \( R_1 \) is regular and \( R_2 \) is regular. Is \( R_1 \cap R_2 \) regular?
- Suppose \( R_1 \) is regular is \( \overline{R_1} \) (complement of \( R_1 \)) regular?
Theorem

\[ L = \{0^n1^n \mid n \geq 0\} = \{\epsilon, 01, 0011, 000111, \ldots\}. \]

*The language* \( L \) *is not a regular language.*