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
- $\emptyset 0$: $\{\}$
- $(\epsilon+1)(01)^*(\epsilon+0)$: 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
- $\emptyset 0$: $\{\}$
- $(\epsilon+1)(01)^*(\epsilon+0)$: 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
- $\emptyset 0$: $\{\}$
- $(\epsilon+1)(01)^*(\epsilon+0)$: 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
- $\emptyset 0$: $\{\}$
- $(\epsilon+1)(01)^*(\epsilon+0)$: 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
- $\emptyset 0$: $\{\}$
- $(\epsilon + 1)(01)^*(\epsilon + 0)$: 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
- $\emptyset 0$: $\{\}$
- $(\epsilon+1)(01)^*(\epsilon+0)$: 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
- $\emptyset 0$: $\{\}$
- $(\epsilon+1)(01)^*(\epsilon+0)$: 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
- $\emptyset 0$: $\{\}$
- $(\epsilon+1)(01)^*(\epsilon+0)$: 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
- $\emptyset 0$: $\{\}$
- $(\epsilon+1)(01)^*(\epsilon+0)$: 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
- $\emptyset 0$: $\{\}$
- $(\epsilon+1)(01)^*(\epsilon+0)$: 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
- $\emptyset 0$: $\{\}$
- $(\epsilon + 1)(01)^*(\epsilon + 0)$: 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: $0^*1\boldsymbol{r}$ where $\boldsymbol{r}$ 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: $0^* + (0^* 1 0^* 1 0^*)^*$
- bitstrings with an odd number of 1's
  one answer: $0^* 1 \boldsymbol{r}$ where $\boldsymbol{r}$ 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: $0^*1\boldsymbol{r}$ where $\boldsymbol{r}$ 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: $0^*1\boldsymbol{r}$ where $\boldsymbol{r}$ is solution to previous part
- bitstrings that do <u>not</u> contain 011 as a substring
- Hard: bitstrings with an odd number of 1s <u>and</u> an odd number of 0s.

Bit strings with odd number of 0s and 1s

The regular expression is

$$
\begin{aligned}
&(00 + 11)^*(01 + 10) \\
&\qquad \Big(00 + 11 + (01 + 10)(00 + 11)^*(01 + 10)\Big)^*
\end{aligned}
$$

(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 on 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 on 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 on 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 on prove an identity?
By induction. On what? Length of r since r is a string obtained from specific inductive rules.

THE END

...

(for now)