CSE 311 Lecture 20: Regular Expressions

Emina Torlak and Kevin Zatloukal
Topics

Structural induction
   A brief review of Lecture 19.

Using structural induction
   Example proofs about recursively defined numbers, strings, and trees.

Regular expressions
   Definition, examples, applications.
Structural induction

A brief review of Lecture 19.
Structural induction proof template

1. Let $P(x)$ be [definition of $P(x)$].
   We will show that $P(x)$ is true for every $x \in S$ by structural induction.

2. Base cases:
   [Proof of $P(s_0), \ldots, P(s_m)$.]

3. Inductive hypothesis:
   Assume that $P(y_0), \ldots, P(y_k)$ are true for some arbitrary $y_0, \ldots, y_k \in S$.

4. Inductive step:
   We want to prove that $P(y)$ is true.
   [Proof of $P(y)$. The proof must invoke the structural inductive hypothesis.]

5. The result follows for all $x \in S$ by structural induction.

Recursive definition of $S$
Basis step:
$s_0 \in S, \ldots, s_m \in S$.
Recursive step:
if $y_0, \ldots, y_k \in S$, then $y \in S$. 
Structural induction proof template

1. Let $P(x)$ be [definition of $P(x)$].
   We will show that $P(x)$ is true for every $x \in S$ by structural induction.
2. Base cases:
   [Proof of $P(s_0), \ldots, P(s_m)$.]
3. Inductive hypothesis:
   Assume that $P(y_0), \ldots, P(y_k)$ are true for some arbitrary $y_0, \ldots, y_k \in S$.
4. Inductive step:
   We want to prove that $P(y)$ is true.
   [Proof of $P(y)$. The proof must invoke the structural inductive hypothesis.]
5. The result follows for all $x \in S$ by structural induction.

Recursive definition of $S$

Basis step:
$s_0 \in S, \ldots, s_m \in S$.

Recursive step:
if $y_0, \ldots, y_k \in S$, then $y \in S$.

If the recursive step of $S$ includes multiple rules for constructing new elements from existing elements, then

3. assume $P$ for the existing elements in every rule, and
4. prove $P$ for the new element in every rule.
Structural induction works just like ordinary induction

1 Let $P(x)$ be [definition of $P(x)$].
   We will show that $P(x)$ is true for every $x \in \mathbb{N}$ by structural induction.

2 Base cases:
   [Proof of $P(0)$.]

3 Inductive hypothesis:
   Assume that $P(n)$ is true for some arbitrary $n \in \mathbb{N}$.

4 Inductive step:
   We want to prove that $P(n + 1)$ is true.
   [Proof of $P(n + 1)$. The proof must invoke the structural inductive hypothesis.]

5 The result follows for all $x \in \mathbb{N}$ by structural induction.

Recursive definition of $\mathbb{N}$
Basis step: $0 \in \mathbb{N}$.
Recursive step:
if $n \in \mathbb{N}$, then $n + 1 \in \mathbb{N}$.

Ordinary induction is just structural induction applied to the recursively defined set of natural numbers!
Understanding structural induction

\[ P(\bullet); \forall L, R \in S. (P(L) \land P(R)) \rightarrow P(\text{Tree}(\bullet, L, R)) \]
\[ \therefore \forall x \in S. P(x) \]

How do we get \( P(\text{Tree}(\bullet, \text{Tree}(\bullet, \bullet, \bullet))) \) from \( P(\bullet) \) and \( \forall L, R \in S. (P(L) \land P(R)) \rightarrow P(\text{Tree}(\bullet, L, R)) \)?

1. First, we have \( \forall L, R \in S. (P(L) \land P(R)) \rightarrow P(\text{Tree}(\bullet, L, R)) \)
2. Next, we have \( P(\bullet) \).
3. Intro \( \land \) on 2 gives us \( P(\bullet) \land P(\bullet) \).
4. Elim \( \forall \) on 1 gives us \( (P(\bullet) \land P(\bullet)) \rightarrow P(\text{Tree}(\bullet, \bullet, \bullet)) \).
5. Modus Ponens on 3 and 4 gives us \( P(\text{Tree}(\bullet, \bullet, \bullet)) \).
6. Intro \( \land \) on 2 and 5 gives us \( P(\bullet) \land P(\text{Tree}(\bullet, \bullet, \bullet)) \).
7. Elim \( \forall \) on 1 gives us
   \[ (P(\bullet) \land P(\text{Tree}(\bullet, \bullet, \bullet)) \rightarrow P(\text{Tree}(\bullet, \text{Tree}(\bullet, \bullet, \bullet)))) \]
8. Modus Ponens on 6 and 7 gives us \( P(\text{Tree}(\bullet, \text{Tree}(\bullet, \bullet, \bullet))) \).
Using structural induction

Example proofs about recursively defined numbers, strings, and trees.
Prove that every $x \in S$ is divisible by 3

Define $S$ by

- **Basis:** $6 \in S$, $15 \in S$.
- **Recursive:** if $x, y \in S$, then $x + y \in S$. 
Prove that every \( x \in S \) is divisible by 3

1. Let \( P(x) \) be \( 3\mid x \).
   We will show that \( P(x) \) is true for every \( x \in S \) by structural induction.

Define \( S \) by
   - Basis: \( 6 \in S, 15 \in S \).
   - Recursive: if \( x, y \in S \), then \( x + y \in S \).
Prove that every $x \in S$ is divisible by 3

1. Let $P(x)$ be $3|x$.
   We will show that $P(x)$ is true for every $x \in S$ by structural induction.
2. Base cases ($x = 6, x = 15$):
   $3|6$ so $P(6)$ holds, and $3|15$ so $P(15)$ holds.

Define $S$ by
   Basis: $6 \in S, 15 \in S$.
   Recursive: if $x, y \in S$, then $x + y \in S$. 
Prove that every $x \in S$ is divisible by 3

1. Let $P(x)$ be $3|x$.
   We will show that $P(x)$ is true for every $x \in S$ by structural induction.

2. **Base cases ($x = 6, x = 15$):**
   - $3|6$ so $P(6)$ holds, and $3|15$ so $P(15)$ holds.

3. **Inductive hypothesis:**
   Assume that $P(x), P(y)$ are true for some arbitrary $x, y \in S$.

---

Define $S$ by

- **Basis:** $6 \in S, 15 \in S$.
- **Recursive:** if $x, y \in S$, then $x + y \in S$. 
Prove that every \( x \in S \) is divisible by 3

1. Let \( P(x) \) be \( 3| x \).  
   We will show that \( P(x) \) is true for every \( x \in S \) by structural induction.

2. Base cases (\( x = 6, x = 15 \)):  
   \( 3|6 \) so \( P(6) \) holds, and \( 3|15 \) so \( P(15) \) holds.

3. Inductive hypothesis:  
   Assume that \( P(x), P(y) \) are true for some arbitrary \( x, y \in S \).

4. Inductive step:  
   We want to prove that \( P(x + y) \) is true.

Define \( S \) by  
Basis: \( 6 \in S, 15 \in S \).  
Recursive: if \( x, y \in S \), then \( x + y \in S \).
Prove that every $x \in S$ is divisible by 3

① Let $P(x)$ be $3|x$.
   We will show that $P(x)$ is true for every $x \in S$ by structural induction.

② Base cases ($x = 6, x = 15$):
   $3|6$ so $P(6)$ holds, and $3|15$ so $P(15)$ holds.

③ Inductive hypothesis:
   Assume that $P(x), P(y)$ are true for some arbitrary $x, y \in S$.

④ Inductive step:
   We want to prove that $P(x + y)$ is true.
   By the inductive hypothesis, $3|x$ and $3|y$, so $x = 3i$ and $y = 3j$ for some $i, j \in \mathbb{Z}$.

Define $S$ by
   Basis: $6 \in S, 15 \in S$.
   Recursive: if $x, y \in S$, then $x + y \in S$. 
Prove that every \( x \in S \) is divisible by 3

1. Let \( P(x) \) be \( 3|x \).
   We will show that \( P(x) \) is true for every \( x \in S \) by structural induction.

2. Base cases (\( x = 6, x = 15 \)):
   \( 3|6 \) so \( P(6) \) holds, and \( 3|15 \) so \( P(15) \) holds.

3. Inductive hypothesis:
   Assume that \( P(x), P(y) \) are true for some arbitrary \( x, y \in S \).

4. Inductive step:
   We want to prove that \( P(x + y) \) is true.
   By the inductive hypothesis, \( 3|x \) and \( 3|y \), so \( x = 3i \) and \( y = 3j \) for some \( i, j \in \mathbb{Z} \). Therefore,
   \( x + y = 3i + 3j = 3(i + j) \) so \( 3|(x + y) \).

Define \( S \) by
   Basis: \( 6 \in S, 15 \in S \).
   Recursive: if \( x, y \in S \), then \( x + y \in S \).
Prove that every \( x \in S \) is divisible by 3

1. Let \( P(x) \) be \( 3|x \).
   We will show that \( P(x) \) is true for every \( x \in S \) by structural induction.

2. Base cases \((x = 6, x = 15)\):
   \( 3|6 \) so \( P(6) \) holds, and \( 3|15 \) so \( P(15) \) holds.

3. Inductive hypothesis:
   Assume that \( P(x), P(y) \) are true for some arbitrary \( x, y \in S \).

4. Inductive step:
   We want to prove that \( P(x + y) \) is true.
   By the inductive hypothesis, \( 3|x \) and \( 3|y \), so \( x = 3i \) and \( y = 3j \) for some \( i, j \in \mathbb{Z} \). Therefore,
   \( x + y = 3i + 3j = 3(i + j) \) so \( 3|(x + y) \). Hence, \( P(x + y) \) is true.

Define \( S \) by
   Basis: \( 6 \in S, 15 \in S \).
   Recursive: if \( x, y \in S \), then \( x + y \in S \).
Prove that every \( x \in S \) is divisible by 3

1. Let \( P(x) \) be \( 3|x \).
   We will show that \( P(x) \) is true for every \( x \in S \) by structural induction.

2. Base cases (\( x = 6, x = 15 \)):
   \( 3|6 \) so \( P(6) \) holds, and \( 3|15 \) so \( P(15) \) holds.

3. Inductive hypothesis:
   Assume that \( P(x), P(y) \) are true for some arbitrary \( x, y \in S \).

4. Inductive step:
   We want to prove that \( P(x + y) \) is true.
   By the inductive hypothesis, \( 3|x \) and \( 3|y \), so \( x = 3i \) and \( y = 3j \) for some \( i, j \in \mathbb{Z} \). Therefore,
   \( x + y = 3i + 3j = 3(i + j) \) so \( 3|(x + y) \). Hence,
   \( P(x + y) \) is true.

5. The result follows for all \( x \in S \) by structural induction.

Define \( S \) by
   Basis: \( 6 \in S, 15 \in S \).
   Recursive: if \( x, y \in S \), then \( x + y \in S \).
Prove \( \text{len}(x \cdot y) = \text{len}(x) + \text{len}(y) \) for all \( x, y \in \Sigma^* \)

Define \( \Sigma^* \) by

Basis: \( \varepsilon \in \Sigma^* \).

Recursive:
if \( w \in \Sigma^* \) and \( a \in \Sigma \),
then \( wa \in \Sigma^* \)

Length
\( \text{len}(\varepsilon) = 0 \)
\( \text{len}(wa) = \text{len}(w) + 1 \)

Concatenation
\( x \cdot \varepsilon = x \)
\( x \cdot (wa) = (x \cdot w)a \)
Prove \(\text{len}(x \cdot y) = \text{len}(x) + \text{len}(y)\) for all \(x, y \in \Sigma^*\)

What object (\(x\) or \(y\)) to do structural induction on?

Define \(\Sigma^*\) by

- **Basis**: \(\epsilon \in \Sigma^*\).
- **Recursive**: if \(w \in \Sigma^*\) and \(a \in \Sigma\), then \(wa \in \Sigma^*\)

**Length**
- \(\text{len}(\epsilon) = 0\)
- \(\text{len}(wa) = \text{len}(w) + 1\)

**Concatenation**
- \(x \cdot \epsilon = x\)
- \(x \cdot (wa) = (x \cdot w)a\)
Prove \( \text{len}(x \cdot y) = \text{len}(x) + \text{len}(y) \) for all \( x, y \in \Sigma^* \)

1. Let \( P(y) \) be \( \forall x \in \Sigma^*. \text{len}(x \cdot y) = \text{len}(x) + \text{len}(y) \).
   We will show that \( P(y) \) is true for every \( y \in \Sigma^* \) by structural induction.

Define \( \Sigma^* \) by
- **Basis**: \( \varepsilon \in \Sigma^* \).
- **Recursive**: if \( w \in \Sigma^* \) and \( a \in \Sigma \), then \( wa \in \Sigma^* \)

**Length**
- \( \text{len}(\varepsilon) = 0 \)
- \( \text{len}(wa) = \text{len}(w) + 1 \)

**Concatenation**
- \( x \cdot \varepsilon = x \)
- \( x \cdot (wa) = (x \cdot w)a \)
Prove $\text{len}(x \cdot y) = \text{len}(x) + \text{len}(y)$ for all $x, y \in \Sigma^*$

1. Let $P(y)$ be $\forall x \in \Sigma^*. \text{len}(x \cdot y) = \text{len}(x) + \text{len}(y)$. We will show that $P(y)$ is true for every $y \in \Sigma^*$ by structural induction.

2. Base case ($y = \varepsilon$):
   Let $x$ in $\Sigma^*$ be arbitrary.

Define $\Sigma^*$ by
- Basis: $\varepsilon \in \Sigma^*$.
- Recursive: if $w \in \Sigma^*$ and $a \in \Sigma$, then $wa \in \Sigma^*$

Length
- $\text{len}(\varepsilon) = 0$
- $\text{len}(wa) = \text{len}(w) + 1$

Concatenation
- $x \cdot \varepsilon = x$
- $x \cdot (wa) = (x \cdot w)a$
Prove $\text{len}(x \cdot y) = \text{len}(x) + \text{len}(y)$ for all $x, y \in \Sigma^*$

1. Let $P(y)$ be $\forall x \in \Sigma^*. \text{len}(x \cdot y) = \text{len}(x) + \text{len}(y)$.
   We will show that $P(y)$ is true for every $y \in \Sigma^*$ by structural induction.

2. **Base case ($y = \varepsilon$):**
   Let $x$ in $\Sigma^*$ be arbitrary. Then, $\text{len}(x \cdot \varepsilon) = \text{len}(x) = \text{len}(x) + \text{len}(\varepsilon)$ since $\text{len}(\varepsilon) = 0$. So $P(\varepsilon)$ is true.

---

Define $\Sigma^*$ by
- **Basis:** $\varepsilon \in \Sigma^*$.
- **Recursive:** if $w \in \Sigma^*$ and $a \in \Sigma$, then $wa \in \Sigma^*$

**Length**
- $\text{len}(\varepsilon) = 0$
- $\text{len}(wa) = \text{len}(w) + 1$

**Concatenation**
- $x \cdot \varepsilon = x$
- $x \cdot (wa) = (x \cdot w)a$
Prove \( \text{len}(x \cdot y) = \text{len}(x) + \text{len}(y) \) for all \( x, y \in \Sigma^* \)

1. Let \( P(y) \) be \( \forall x \in \Sigma^*. \text{len}(x \cdot y) = \text{len}(x) + \text{len}(y) \).
   We will show that \( P(y) \) is true for every \( y \in \Sigma^* \) by structural induction.

2. Base case \( (y = \varepsilon) \):
   Let \( x \) in \( \Sigma^* \) be arbitrary. Then, \( \text{len}(x \cdot \varepsilon) = \text{len}(x) = \text{len}(x) + \text{len}(\varepsilon) \)
   since \( \text{len}(\varepsilon) = 0 \). So \( P(\varepsilon) \) is true.

3. Inductive hypothesis:
   Assume that \( P(w) \) is true for some arbitrary \( w \in \Sigma^* \).

Define \( \Sigma^* \) by
   Basis: \( \varepsilon \in \Sigma^* \).
   Recursive:
   if \( w \in \Sigma^* \) and \( a \in \Sigma \),
   then \( wa \in \Sigma^* \).

Length
   \( \text{len}(\varepsilon) = 0 \)
   \( \text{len}(wa) = \text{len}(w) + 1 \)

Concatenation
   \( x \cdot \varepsilon = x \)
   \( x \cdot (wa) = (x \cdot w)a \)
Prove $\operatorname{len}(x \cdot y) = \operatorname{len}(x) + \operatorname{len}(y)$ for all $x, y \in \Sigma^*$

1. Let $P(y)$ be $\forall x \in \Sigma^*. \operatorname{len}(x \cdot y) = \operatorname{len}(x) + \operatorname{len}(y)$.
   We will show that $P(y)$ is true for every $y \in \Sigma^*$ by structural induction.

2. Base case ($y = \varepsilon$):
   Let $x$ in $\Sigma^*$ be arbitrary. Then, $\operatorname{len}(x \cdot \varepsilon) = \operatorname{len}(x) = \operatorname{len}(x) + \operatorname{len}(\varepsilon)$
   since $\operatorname{len}(\varepsilon) = 0$. So $P(\varepsilon)$ is true.

3. Inductive hypothesis:
   Assume that $P(w)$ is true for some arbitrary $w \in \Sigma^*$.

4. Inductive step:
   We want to prove that $P(wa)$ is true for every $a \in \Sigma$.

Define $\Sigma^*$ by
Basis: $\varepsilon \in \Sigma^*$.
Recursive:
if $w \in \Sigma^*$ and $a \in \Sigma$,
then $wa \in \Sigma^*$

Length
$\operatorname{len}(\varepsilon) = 0$
$\operatorname{len}(wa) = \operatorname{len}(w) + 1$

Concatenation
$x \cdot \varepsilon = x$
$x \cdot (wa) = (x \cdot w)a$
Prove $\text{len}(x \cdot y) = \text{len}(x) + \text{len}(y)$ for all $x, y \in \Sigma^*$

1. Let $P(y)$ be $\forall x \in \Sigma^*. \text{len}(x \cdot y) = \text{len}(x) + \text{len}(y)$. We will show that $P(y)$ is true for every $y \in \Sigma^*$ by structural induction.

2. Base case ($y = \varepsilon$):
   Let $x$ in $\Sigma^*$ be arbitrary. Then, $\text{len}(x \cdot \varepsilon) = \text{len}(x) = \text{len}(x) + \text{len}(\varepsilon)$ since $\text{len}(\varepsilon) = 0$. So $P(\varepsilon)$ is true.

3. Inductive hypothesis:
   Assume that $P(w)$ is true for some arbitrary $w \in \Sigma^*$.

4. Inductive step:
   We want to prove that $P(wa)$ is true for every $a \in \Sigma$.
   Let $a \in \Sigma$ and $x \in \Sigma^*$ be arbitrary. Then

Define $\Sigma^*$ by

Basis: $\varepsilon \in \Sigma^*$.
Recursive:
if $w \in \Sigma^*$ and $a \in \Sigma$, then $wa \in \Sigma^*$

Length
$\text{len}(\varepsilon) = 0$
$\text{len}(wa) = \text{len}(w) + 1$

Concatenation
$x \cdot \varepsilon = x$
$x \cdot (wa) = (x \cdot w)a$
Prove \( \text{len}(x \bullet y) = \text{len}(x) + \text{len}(y) \) for all \( x, y \in \Sigma^* \)

1. Let \( P(y) \) be \( \forall x \in \Sigma^*. \text{len}(x \bullet y) = \text{len}(x) + \text{len}(y) \).
   We will show that \( P(y) \) is true for every \( y \in \Sigma^* \) by structural induction.

2. Base case \( (y = \varepsilon) \):
   Let \( x \) in \( \Sigma^* \) be arbitrary. Then, \( \text{len}(x \bullet \varepsilon) = \text{len}(x) = \text{len}(x) + \text{len}(\varepsilon) \) since \( \text{len}(\varepsilon) = 0 \). So \( P(\varepsilon) \) is true.

3. Inductive hypothesis:
   Assume that \( P(w) \) is true for some arbitrary \( w \in \Sigma^* \).

4. Inductive step:
   We want to prove that \( P(wa) \) is true for every \( a \in \Sigma \).
   Let \( a \in \Sigma \) and \( x \in \Sigma^* \) be arbitrary. Then
   \[
   \text{len}(x \bullet wa) = \text{len}((x \bullet wa) \bullet a)
   \]
   \[
   = \text{len}(x \bullet w) + 1 \quad \text{by defn of } \bullet
   \]
   \[
   = \text{len}(x) + \text{len}(w) + 1 \quad \text{by defn of len}
   \]
   \[
   = \text{len}(x) + \text{len}(wa) \quad \text{by IH}
   \]
   \[
   = \text{len}(x) + \text{len}(wa) \quad \text{by defn of len}
   \]

Define \( \Sigma^* \) by
- Basis: \( \varepsilon \in \Sigma^* \).
- Recursive:
  - if \( w \in \Sigma^* \) and \( a \in \Sigma \), then \( wa \in \Sigma^* \)

Length
\[
\text{len}(\varepsilon) = 0
\]
\[
\text{len}(wa) = \text{len}(w) + 1
\]

Concatenation
\[
x \bullet \varepsilon = x
\]
\[
x \bullet (wa) = (x \bullet w)a
\]
Prove \( \text{len}(x \cdot y) = \text{len}(x) + \text{len}(y) \) for all \( x, y \in \Sigma^* \)

1. Let \( P(y) \) be \( \forall x \in \Sigma^*. \text{len}(x \cdot y) = \text{len}(x) + \text{len}(y) \).
   We will show that \( P(y) \) is true for every \( y \in \Sigma^* \) by structural induction.

2. Base case \( (y = \varepsilon) \):
   Let \( x \) in \( \Sigma^* \) be arbitrary. Then, \( \text{len}(x \cdot \varepsilon) = \text{len}(x) = \text{len}(x) + \text{len}(\varepsilon) \)
   since \( \text{len}(\varepsilon) = 0 \). So \( P(\varepsilon) \) is true.

3. Inductive hypothesis:
   Assume that \( P(w) \) is true for some arbitrary \( w \in \Sigma^* \).

4. Inductive step:
   We want to prove that \( P(wa) \) is true for every \( a \in \Sigma \).
   Let \( a \in \Sigma \) and \( x \in \Sigma^* \) be arbitrary. Then
   \[
   \text{len}(x \cdot wa) = \text{len}((x \cdot wa)) \quad \text{by defn of } \cdot \\
   = \text{len}(x \cdot w) + 1 \quad \text{by defn of } \text{len} \\
   = \text{len}(x) + \text{len}(w) + 1 \quad \text{by IH} \\
   = \text{len}(x) + \text{len}(wa) \quad \text{by defn of } \text{len} \\
   \]
   so \( \text{len}(x \cdot wa) = \text{len}(x) + \text{len}(wa) \) for all \( x \in \Sigma^* \), and \( P(wa) \) is true.
Prove \( \text{len}(x \cdot y) = \text{len}(x) + \text{len}(y) \) for all \( x, y \in \Sigma^* \)

1. Let \( P(y) \) be \( \forall x \in \Sigma^*. \text{len}(x \cdot y) = \text{len}(x) + \text{len}(y) \).
   We will show that \( P(y) \) is true for every \( y \in \Sigma^* \) by structural induction.

2. **Base case** (\( y = \epsilon \)):
   Let \( x \) in \( \Sigma^* \) be arbitrary. Then, \( \text{len}(x \cdot \epsilon) = \text{len}(x) = \text{len}(x) + \text{len}(\epsilon) \)
   since \( \text{len}(\epsilon) = 0 \). So \( P(\epsilon) \) is true.

3. **Inductive hypothesis**:
   Assume that \( P(w) \) is true for some arbitrary \( w \in \Sigma^* \).

4. **Inductive step**:
   We want to prove that \( P(wa) \) is true for every \( a \in \Sigma \).
   Let \( a \in \Sigma \) and \( x \in \Sigma^* \) be arbitrary. Then
   \[
   \text{len}(x \cdot wa) = \text{len}((x \cdot wa)) \quad \text{by defn of } \cdot \\
   = \text{len}(x \cdot w) + 1 \quad \text{by defn of len} \\
   = \text{len}(x) + \text{len}(w) + 1 \quad \text{by IH} \\
   = \text{len}(x) + \text{len}(wa) \quad \text{by defn of len}
   \]
   So \( \text{len}(x \cdot wa) = \text{len}(x) + \text{len}(wa) \) for all \( x \in \Sigma^* \), and \( P(wa) \) is true.

5. The result follows for all \( y \in \Sigma^* \) by structural induction.
Prove $|t| \leq 2^{|t|} + 1 - 1$ for every rooted binary tree $t$

Define $S$ by

Basis: $\bullet \in S$.

Recursive:
if $L, R \in S$, then
$\text{Tree}(\bullet, L, R) \in S$

Size
$|\bullet| = 1$
$|\text{Tree}(\bullet, L, R)| = 1 + |L| + |R|

Height
$|\bullet| = 0$
$|\text{Tree}(\bullet, L, R)| = 1 + \max(|L|, |R|)$
Prove $|t| \leq 2^{|t|+1} - 1$ for every rooted binary tree $t$

1. Let $P(t)$ be $|t| \leq 2^{|t|+1} - 1$.

   We will show that $P(t)$ is true for every $t \in S$ by structural induction.

Define $S$ by

**Basis:** $\bullet \in S$.

**Recursive:**

- if $L, R \in S$, then $\text{Tree}(\bullet, L, R) \in S$

**Size**

$|\bullet| = 1$

$|\text{Tree}(\bullet, L, R)| = 1 + |L| + |R|$

**Height**

$[\bullet] = 0$

$[\text{Tree}(\bullet, L, R)] = 1 + \max([L], [R])$
Prove $|t| \leq 2^{[t]+1} - 1$ for every rooted binary tree $t$

1. Let $P(t)$ be $|t| \leq 2^{[t]+1} - 1$.
   
   We will show that $P(t)$ is true for every $t \in S$ by structural induction.

2. Base case ($t = \bullet$):
   
   $|\bullet| = 1 = 2^1 - 1 = 2^{0+1} - 1 = 2^{[\bullet]+1} - 1$ so $P(\bullet)$ is true.

Define $S$ by

Basis: $\bullet \in S$.
Recursive: if $L, R \in S$, then $\text{Tree}(\bullet, L, R) \in S$

Size

$|\bullet| = 1$
$|\text{Tree}(\bullet, L, R)| = 1 + |L| + |R|$

Height

$[\bullet] = 0$
$[[\text{Tree}(\bullet, L, R)] = 1 + \max([L], [R])$
Prove \( |t| \leq 2^{|t|+1} - 1 \) for every rooted binary tree \( t \)

1. Let \( P(t) \) be \( |t| \leq 2^{|t|+1} - 1 \).
   We will show that \( P(t) \) is true for every \( t \in S \) by structural induction.

2. Base case (\( t = \cdot \)):
   \[ |\cdot| = 1 = 2^1 - 1 = 2^{0+1} - 1 = 2^{\lfloor \cdot \rfloor+1} - 1 \] so \( P(\cdot) \) is true.

3. Inductive hypothesis:
   Assume that \( P(L) \) and \( P(R) \) are true for some arbitrary \( L, R \in S \).

Define \( S \) by

- Basis: \( \cdot \in S \).
- Recursive:
  - if \( L, R \in S \), then \( \text{Tree}(\cdot, L, R) \in S \).

Size

\[ |\cdot| = 1 \]
\[ |\text{Tree}(\cdot, L, R)| = 1 + |L| + |R| \]

Height

\[ |\cdot| = 0 \]
\[ |\text{Tree}(\cdot, L, R)| = 1 + \max(|L|, |R|) \]
Prove $|t| \leq 2^{[t]+1} - 1$ for every rooted binary tree $t$

1. Let $P(t)$ be $|t| \leq 2^{[t]+1} - 1$.
   We will show that $P(t)$ is true for every $t \in S$ by structural induction.

2. Base case ($t = \bullet$):
   
   $|\bullet| = 1 = 2^1 - 1 = 2^{0+1} - 1 = 2^{[\bullet]+1} - 1$ so $P(\bullet)$ is true.

3. Inductive hypothesis:
   Assume that $P(L)$ and $P(R)$ are true for some arbitrary $L, R \in S$.

4. Inductive step:
   We want to prove that $P(\text{Tree}(\bullet, L, R))$ is true.
Prove $|t| \leq 2^{[t]+1} - 1$ for every rooted binary tree $t$

1. Let $P(t)$ be $|t| \leq 2^{[t]+1} - 1$.
   We will show that $P(t)$ is true for every $t \in S$ by structural induction.

2. Base case ($t = \bullet$):
   $| \cdot | = 1 = 2^1 - 1 = 2^{0+1} - 1 = 2^{[\bullet]+1} - 1$ so $P(\bullet)$ is true.

3. Inductive hypothesis:
   Assume that $P(L)$ and $P(R)$ are true for some arbitrary $L, R \in S$.

4. Inductive step:
   We want to prove that $P(\text{Tree}(\bullet, L, R))$ is true.
   
   $|\text{Tree}(\bullet, L, R)| = 1 + |L| + |R|$
   $\leq 1 + (2^{[L]+1} - 1) + (2^{[R]+1} - 1)$
   $\leq 2^{[L]+1} + 2^{[R]+1} - 1$
   $\leq 2(2^{\max([L],[R])+1}) - 1$
   $= 2(2^{\lceil [\text{Tree}(\bullet, L, R)] \rceil}) - 1$
   $= 2^{\lceil [\text{Tree}(\bullet, L, R)] \rceil + 1} - 1$

   by defn of $| |$
   by IH
   algebra
   by defn of max
   by defn of $\lceil \rceil$

   which is the desired result.

Define $S$ by

Basis: $\bullet \in S$.
Recursive: if $L, R \in S$, then $\text{Tree}(\bullet, L, R) \in S$

Size
$|\bullet| = 1$
$|\text{Tree}(\bullet, L, R)| = 1 + |L| + |R|$

Height
$[\bullet] = 0$
$\lceil [\text{Tree}(\bullet, L, R)] \rceil = 1 + \max([L],[R])$
Prove $|t| \leq 2^{\lceil t \rceil} + 1 - 1$ for every rooted binary tree $t$

1. Let $P(t)$ be $|t| \leq 2^{\lceil t \rceil} + 1 - 1$.
   
   We will show that $P(t)$ is true for every $t \in S$ by structural induction.

2. Base case ($t = \bullet$):
   
   $|\bullet| = 1 = 2^1 - 1 = 2^{0+1} - 1 = 2^{\lceil \bullet \rceil} + 1 - 1$ so $P(\bullet)$ is true.

3. Inductive hypothesis:
   
   Assume that $P(L)$ and $P(R)$ are true for some arbitrary $L, R \in S$.

4. Inductive step:
   
   We want to prove that $P(\text{Tree}(\bullet, L, R))$ is true.
   
   $|\text{Tree}(\bullet, L, R)| = 1 + |L| + |R| = 1 + (2^{\lceil L \rceil} + 1) + (2^{\lceil R \rceil} + 1) - 1$
   
   $\leq 1 + 2^{\lceil L \rceil} + 2^{\lceil R \rceil} + 1 - 1$
   
   $\leq 2^{\lceil L \rceil + 1} + 2^{\lceil R \rceil + 1} - 1$
   
   $\leq 2(2^{\max(\lceil L \rceil, \lceil R \rceil) + 1}) - 1$
   
   $= 2(2^{\lceil \text{Tree}(\bullet, L, R) \rceil}) - 1$
   
   $= 2^{\lceil \text{Tree}(\bullet, L, R) \rceil + 1} - 1$
   
   which is the desired result.

5. The result follows for all $t \in S$ by structural induction.
Regular expressions
Definition, examples, applications.
Sets of strings as languages

A *language* is a sets of strings with specific syntax, e.g.:
- Syntactically correct Java/C/C++ programs.
- The set $\Sigma^*$ of all strings over the alphabet $\Sigma$.
- Palindromes over $\Sigma$.
- Binary strings with no 1’s before 0’s.
Sets of strings as languages

A language is a sets of strings with specific syntax, e.g.:
- Syntactically correct Java/C/C++ programs.
- The set $\Sigma^*$ of all strings over the alphabet $\Sigma$.
- Palindromes over $\Sigma$.
- Binary strings with no 1’s before 0’s.

Regular expressions let us specify regular languages, e.g.:
- All binary strings.
- The strings \{0000, 0010, 1000, 1010\}.
- All strings that contain the string “CSE311”.

Regular expressions over $\Sigma$: syntax

Basis step:
- $\emptyset, \epsilon$ are regular expressions.
- $a$ is a regular expression for any $a \in \Sigma$.

Recursive step:
- If $A$ and $B$ are regular expressions, then so are $AB, A \cup B$, and $A^*$.
Regular expressions over $\Sigma$: syntax

Basis step:
- $\emptyset, \varepsilon$ are regular expressions.
- $a$ is a regular expression for any $a \in \Sigma$.

Recursive step:
- If $A$ and $B$ are regular expressions, then so are $AB, A \cup B$, and $A^*$.

Examples: regular expressions of $\Sigma = \{0, 1\}$
- Basis: $\emptyset, \varepsilon, 0, 1$.
- Recursive: $01011, 0^*1^*, (0 \cup 1)0(0 \cup 1)0$, etc.
Regular expressions over $\Sigma$: semantics

A regular expression over $\Sigma$ represents a set of strings over $\Sigma$. 
Regular expressions over $\Sigma$: semantics

A regular expression over $\Sigma$ represents a set of strings over $\Sigma$. $\emptyset$ represents the set with no strings.
Regular expressions over $\Sigma$: semantics

A regular expression over $\Sigma$ represents a set of strings over $\Sigma$.

- $\emptyset$ represents the set with no strings.
- $\varepsilon$ represents the set $\{\varepsilon\}$. 
Regular expressions over $\Sigma$: semantics

A regular expression over $\Sigma$ represents a set of strings over $\Sigma$.

- $\emptyset$ represents the set with no strings.
- $\varepsilon$ represents the set $\{\varepsilon\}$.
- $a$ represents the set $\{a\}$.
Regular expressions over $\Sigma$: semantics

A regular expression over $\Sigma$ represents a set of strings over $\Sigma$.

- $\emptyset$ represents the set with no strings.
- $\epsilon$ represents the set $\{\epsilon\}$.
- $a$ represents the set $\{a\}$.
- $AB$ represents the concatenation of the sets represented by $A$ and $B$: $\{a \cdot b \mid a \in A, b \in B\}$. 
Regular expressions over $\Sigma$: semantics

A regular expression over $\Sigma$ represents a set of strings over $\Sigma$.

$\emptyset$ represents the set with no strings.

$\epsilon$ represents the set $\{\epsilon\}$.

$a$ represents the set $\{a\}$.

$AB$ represents the concatenation of the sets represented by $A$ and $B$: $\{a \cdot b \mid a \in A, b \in B\}$.

$A \cup B$ represents the union of the sets represented by $A$ and $B$: $A \cup B$. 
Regular expressions over $\Sigma$: semantics

A regular expression over $\Sigma$ represents a set of strings over $\Sigma$.

- $\emptyset$ represents the set with no strings.
- $\varepsilon$ represents the set $\{\varepsilon\}$.
- $a$ represents the set $\{a\}$.
- $AB$ represents the concatenation of the sets represented by $A$ and $B$: $\{a \cdot b \mid a \in A, b \in B\}$.
- $A \cup B$ represents the union of the sets represented by $A$ and $B$: $A \cup B$.
- $A^*$ represents the concatenation of the set represented by $A$ with itself zero or more times: $\bigcup_{k=0}^{\infty} A^k$, where $A^0 = \{\varepsilon\}$, $A^1 = A$, $A^2 = (AA)$, etc.
Regular expressions over $\Sigma$: semantics

A regular expression over $\Sigma$ represents a set of strings over $\Sigma$.

- $\emptyset$ represents the set with no strings.
- $\varepsilon$ represents the set $\{\varepsilon\}$.
- $a$ represents the set $\{a\}$.
- $AB$ represents the concatenation of the sets represented by $A$ and $B$: $\{a \cdot b \mid a \in A, b \in B\}$.
- $A \cup B$ represents the union of the sets represented by $A$ and $B$: $A \cup B$.
- $A^*$ represents the concatenation of the set represented by $A$ with itself zero or more times: $\bigcup_{k=0}^{\infty} A^k$, where $A^0 = \{\varepsilon\}, A^1 = A, A^2 = (AA), \text{ etc.}$

Regular expressions can also be viewed as “patterns.” A regular expression $R$ matches a string $s$ if $s$ is a member of the set of strings represented by $R$. 
Examples of regular expressions

001*

0*1*

(0 ∪ 1)0(0 ∪ 1)0

(0*1*)*

(0 ∪ 1)*0110(0 ∪ 1)*
Examples of regular expressions

001*

Binary strings with “00” followed by any number of 1s.

0*1*

(0 ∪ 1)0(0 ∪ 1)0

(0*1*)*

(0 ∪ 1)*0110(0 ∪ 1)*
Examples of regular expressions

001*
   Binary strings with “00” followed by any number of 1s.
0*1*
   Binary strings with any number of 0s followed by any number of 1s.
(0 ∪ 1)0(0 ∪ 1)0

(0*1*)*

(0 ∪ 1)*0110(0 ∪ 1)*
Examples of regular expressions

001*
   Binary strings with “00” followed by any number of 1s.

0*1*
   Binary strings with any number of 0s followed by any number of 1s.

(0 ∪ 1)0(0 ∪ 1)0
   \{0000, 0010, 1000, 1010\}

(0*1*)*

(0 ∪ 1)*0110(0 ∪ 1)*
Examples of regular expressions

001*
  Binary strings with “00” followed by any number of 1s.

0*1*
  Binary strings with any number of 0s followed by any number of 1s.

(0 ∪ 1)0(0 ∪ 1)0
  {0000, 0010, 1000, 1010}

(0*1*)*
  All binary strings.

(0 ∪ 1)*0110(0 ∪ 1)*
Examples of regular expressions

\(001^*\)

Binary strings with “00” followed by any number of 1s.

\(0^*1^*\)

Binary strings with any number of 0s followed by any number of 1s.

\((0 \cup 1)0(0 \cup 1)0\)

\{0000, 0010, 1000, 1010\}

\((0^*1^*)^*\)

All binary strings.

\((0 \cup 1)^*0110(0 \cup 1)^*\)

Binary strings that contain “0110”.
Regular expressions in practice

Used to define the *tokens* in a programming language.
   Legal variable names, keywords, etc.

Used in *grep*, a Unix program that searches for patterns in a set of files.
   For example, `grep "311" *.md` searches for the string “311” in all Markdown files in the current directory.

Used in programs to process strings.
   These slides are generated with the help of regular expressions :(
Summary

Use structural induction to prove properties of recursive structures.
Follows from ordinary induction but is easier to use.
As powerful as ordinary induction.

To prove $\forall x \in S. P(x)$ using structural induction:
Show that $P$ holds for the elements in the basis step of $S$.
Assume $P$ for every existing element of $S$ named in the recursive step.
Prove $P$ for every new element of $S$ created in the recursive step.

A regular expression defines a set of strings over an alphabet $\Sigma$.
$\emptyset, \varepsilon$, and $a \in \Sigma$ are regular expressions.
If $A$ and $B$ are regular expressions, then so are $(AB), (A \cup B), A^*$. Many practical applications, from grep to everyday programming.