Spring 2015
Lecture 19: Structural induction and regular expressions

WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.

OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!

BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS! IT'S HOPELESS!

EVERYBODY STAND BACK.

I KNOW REGULAR EXPRESSIONS.

EXPLoding the Day.
• An *alphabet* $\Sigma$ is any finite set of characters.

e.g. $\Sigma = \{0,1\}$ or $\Sigma = \{A, B, C, \ldots, X, Y, Z\}$ or

$\Sigma = \ldots$

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>28</td>
<td>95</td>
<td>153</td>
<td>186</td>
<td>219</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>29</td>
<td>96</td>
<td>154</td>
<td>187</td>
<td>220</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>97</td>
<td>155</td>
<td>188</td>
<td>221</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>31</td>
<td>98</td>
<td>156</td>
<td>189</td>
<td>222</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>99</td>
<td>157</td>
<td>190</td>
<td>223</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>33</td>
<td>100</td>
<td>158</td>
<td>191</td>
<td>224</td>
<td>225</td>
<td>226</td>
</tr>
<tr>
<td>7</td>
<td>34</td>
<td>101</td>
<td>159</td>
<td>192</td>
<td>224</td>
<td>225</td>
<td>226</td>
</tr>
<tr>
<td>8</td>
<td>35</td>
<td>102</td>
<td>160</td>
<td>193</td>
<td>226</td>
<td>227</td>
<td>226</td>
</tr>
<tr>
<td>9</td>
<td>36</td>
<td>103</td>
<td>161</td>
<td>194</td>
<td>227</td>
<td>228</td>
<td>228</td>
</tr>
<tr>
<td>10</td>
<td>37</td>
<td>104</td>
<td>162</td>
<td>195</td>
<td>228</td>
<td>228</td>
<td>228</td>
</tr>
<tr>
<td>11</td>
<td>38</td>
<td>105</td>
<td>163</td>
<td>196</td>
<td>229</td>
<td>229</td>
<td>229</td>
</tr>
</tbody>
</table>

• The set $\Sigma^*$ of *strings* over the alphabet $\Sigma$ is defined by
  – **Basis:** $\mathcal{E} \in \Sigma^*$ ($\mathcal{E}$ is the empty string)
  – **Recursive:** if $w \in \Sigma^*$, $a \in \Sigma$, then $wa \in \Sigma^*$
Function definitions on recursively defined sets

**Length:**

\[
\text{len } (\varepsilon) = 0; \\
\text{len } (wa) = 1 + \text{len}(w); \text{ for } w \in \Sigma^*, a \in \Sigma
\]

**Reversal:**

\[
\varepsilon^R = \varepsilon \\
(wa)^R = aw^R \text{ for } w \in \Sigma^*, a \in \Sigma
\]

**Concatenation:**

\[
x \cdot \varepsilon = x \text{ for } x \in \Sigma^* \\
x \cdot wa = (x \cdot w)a \text{ for } x, w \in \Sigma^*, a \in \Sigma
\]
How to prove $\forall x \in S, P(x)$ is true:

**Base Case:** Show that $P(u)$ is true for all specific elements $u$ of $S$ mentioned in the *Basis step*

**Inductive Hypothesis:** Assume that $P$ is true for some arbitrary values of *each* of the existing named elements mentioned in the *Recursive step*

**Inductive Step:** Prove that $P(w)$ holds for each of the new elements $w$ constructed in the *Recursive step* using the named elements mentioned in the Inductive Hypothesis

**Conclude** that $\forall x \in S, P(x)$
Structural induction for strings

Let $S$ be a set of strings over $\Sigma = \{a, b\}$ defined by

**Basis:** $a \in S$

**Recursive:**
- If $w \in S$ then $wa \in S$ and $wba \in S$
- If $u, v \in S$ then $uv \in S$

**Claim:** If $w \in S$ then $w$ has more $a$'s than $b$'s.

**Base case:** $a$ has more $a$'s than $b$'s so $P(a)$ holds.

**IH:** Assume $P(w), P(u), P(v)$ for some $w, u, v \in S$.

- If $\#a(w) > \#b(w)$ then $\#a(wa) > \#b(wa) \Rightarrow P(wa)$
- If $\#b(w) > \#a(w)$ then $\#a(wba) > \#b(wba) \Rightarrow P(wba)$
prove: \( \text{len}(x \cdot y) = \text{len}(x) + \text{len}(y) \) for all \( x, y \in \Sigma^* \)

Let \( P(y) \) be "\( \text{len}(x \cdot y) = \text{len}(x) + \text{len}(y) \) for all \( x \in \Sigma^* \)"

Goal: \( P(y) \) \( \forall y \in \Sigma^* \)

Base case: \( \forall x \in \Sigma^* \), \( \text{len}(x \cdot \varepsilon) = \text{len}(x) = \text{len}(x) + \text{len}(\varepsilon) \) \( \Rightarrow P(\varepsilon) \)

II: Assume \( P(w) \) for some \( w \in \Sigma^* \).

III: Fix \( x \in \Sigma^* \).

\[
\begin{align*}
\text{len}(x \cdot (wa)) &= \text{len}((x \cdot w)a) = 1 + \text{len}(x \cdot w) \\
&= 1 + \text{len}(x) + \text{len}(w) \\
&= \text{len}(x) + \text{len}(wa)
\end{align*}
\]

By structure \( P(y) \).

Length:
\[
\begin{align*}
\text{len}(\varepsilon) &= 0; \\
\text{len}(wa) &= 1 + \text{len}(w); \text{ for } w \in \Sigma^*, a \in \Sigma
\end{align*}
\]
• **Basis:** is a rooted binary tree

• **Recursive step:**

\[
\text{If } T_1 \text{ and } T_2 \text{ are rooted binary trees, then so is:}
\]

\[
\begin{aligned}
T_1 & \quad \text{and} \quad T_2 \\
\end{aligned}
\]
defining a function on rooted binary trees

- $\text{size}(\cdot) = 1$

- \[
\text{size}\left(\begin{array}{c}
T_1 \\
\end{array}\right) = 1 + \text{size}(T_1) + \text{size}(T_2)
\]

- $\text{height}(\cdot) = 0$

- \[
\text{height}\left(\begin{array}{c}
T_1 \\
\end{array}\right) = 1 + \max\{\text{height}(T_1), \text{height}(T_2)\}
\]
**Claim:** For every rooted binary tree $T$, $\text{size}(T) \leq 2^{\text{height}(T)+1} - 1$

\[
\Pr(T) = \text{"size}(T) \leq 2^{\text{height}(T)+1} - 1"
\]

**Base case:**

\[
\text{size}(\cdot) = 1 = 2^{0+1} - 1
\]

\[
= 2^{\text{height}(\cdot)+1} - 1 \Rightarrow \Pr(\cdot) \Rightarrow \Pr(T)
\]

**IH:** $\Pr(T_1)$ and $\Pr(T_2)$ for some RBTs $T_1$, $T_2$

**Goal:** $\Pr(T_3)$

\[
\text{size}(T_3) = 1 + \text{size}(T_1) + \text{size}(T_2)
\]

\[
\leq 1 + (2^{h(T_1)+1} - 1) + (2^{h(T_2)+1} - 1)
\]

\[
= 2(2^{h(T_1)} + 2^{h(T_2)}) - 1 \leq 2 \cdot 2 \cdot 2 - 1
\]
Sets of strings that satisfy special properties are called languages.

Examples:

- English sentences
- Syntactically correct Java/C/C++ programs
- $\Sigma^* = \text{All strings over alphabet } \Sigma$
- Palindromes over $\Sigma$
- Binary strings that don’t have a 0 after a 1
- Legal variable names, keywords in Java/C/C++
- Binary strings with an equal # of 0’s and 1’s