Strings

- finite sequence of symbols/tokens
- over an alphabet $\Sigma$

The empty string is denoted $\epsilon$

*Concatenation* of $x$ and $y$ is written $xy$

A *language* is a set of strings
Operations on languages

usual set operations

concatenation of languages

\[ LM = \{xy \mid x \in L, y \in M\} \]

Kleene closure

\[ L^* = \{x_1 \ldots x_k \mid k \geq 0, x_1, \ldots, x_k \in L\} \]

\[ \epsilon \in L^* \ (k = 0) \]
Kleene closure

$L^*$

• $\epsilon \in L^*$

• if $x \in L$ then $x \in L^*$

• if $x \in L^*$ and $y \in L^*$ then $xy \in L^*$

• nothing else is in $L^*$

$L^+ = \{ x_1 \ldots x_k \mid k \geq 1, x_1, \ldots, x_k \in L \}$

• $\epsilon$ need not be in $L^+$
Regular expressions

to specify languages/patterns over an alphabet $\Sigma$

Building blocks:

- $\emptyset$, denotes the empty language
- $\epsilon$, denotes $\{\epsilon\}$
- every symbol $a \in \Sigma$
- operations concatenation, $\cup$ and $*$

operator precedence: $* \succ$ concatenation $\succ \cup$
Regular expressions

Examples:

$$(0 \cup 1)^*0$$  binary strings ending in 0

$$(0 \cup 1)^*00(0 \cup 1)^*$$  binary strings that contain two consecutive 0’s

$$0^*01^*1$$  strings of the form one or more 0’s followed by one or more 1’s
In Egrep and Perl

<table>
<thead>
<tr>
<th>Classical notation</th>
<th>egrep/perl notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \cup y$</td>
<td>$x \mid y$</td>
</tr>
<tr>
<td>$x^*$</td>
<td>$x^*$</td>
</tr>
</tbody>
</table>

looks for substrings to match

eve% echo "aabb" | egrep 'ab'
aabb
eve% echo "aabb" | egrep 'ba'
eve%
Anchors in egrep and perl

<table>
<thead>
<tr>
<th>character</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td>position at the start of the line</td>
</tr>
<tr>
<td>$</td>
<td>position at the end of the line</td>
</tr>
</tbody>
</table>

eve% echo "aabb" | egrep ’^ab’
eve% echo "aabb" | egrep ’^a*b’
aabb
eve% echo "aabb" | egrep ’^a*b$’
eve% echo "aabb" | egrep ’^a*b*$’
aabb
eve% echo "aabb" | egrep ’^a* b*$’
eve%