RegExs with DFA and Parse Trees

CS230 Tutorial 11
Regular Expressions (Regex)

This way of representing regular languages using metacharacters. Here are some of the most important ones to know:

- **|** -- OR
  - example: a | b -- this represents a OR b
- **()** -- grouping, basically like regular parentheses in math
  - example: m(i | a)lk -- this represents ‘milk’ OR ‘malk’
- **[]** -- matches one character in the list specified in the brackets
  - example: [abc] -- this represents a OR b OR c
  - can use - to specify a range in the brackets, i.e. [a-c] is the same as [abc]
- **^** -- NOT
- **.** -- any ONE character
  - example: a.c -- this represents a[every ascii character]c
More metacharacters...

- `?` -- the previous character 0 or 1 time
  - example: `aa?` -- this means a OR aa
- `+` -- the previous character 1 or more times
  - example: `a+` -- this matches a OR aa OR aaa OR aaaa OR ….. infinite a’s
- `*` -- the previous character 0 or more times
  - example: `a*` -- this matches Φ OR a OR aa OR aaa OR …… infinite a’s
- `\` -- escape: you use this when you want to treat one of the metacharacters as a normal ASCII character
  - example: `a\*` -- this means a* literally, not aa or aaa or …
  - example: `\` -- this means the backslash character
Regex Examples

Make a regex for the language of any combination of the letters a and b.

Make a regex for the language of any number of a’s, followed by an even number of c’s.

Describe the language (with an english sentence) represented by: a+[0-9]*xy?
Make a regex for the language of any combination of the letters a and b.

**Solution:** (a | b)* or [ab]*

Make a regex for the language of any number of a’s, followed by an even number of c’s.

**Solution:** a*(cc)*

Describe the language (with an english sentence) represented by: a+[0-9]*xy?

**Solution:** 1 or more a’s, followed by any number of digits (remember, digit is any number from 0-9), followed by x, and then 0 or 1 y.
Regex Practice

Make a regex for the language of at least 2 b’s, followed by an a, b, or c, followed by two a’s.

Make a regex for the language of any number of a’s, followed by a c or two b’s, followed by an odd number of c’s and then an optional a.

Describe the language (with an english sentence) represented by: a+b?c+b?d*
Regex Practice Solutions

Make a regex for the language of at least 2 b’s, followed by an a, b, or c, followed by two a’s.

**Solution:** `bb+[abc]aa`

Make a regex for the language of any number of a’s, followed by a c or two b’s, followed by an odd number of c’s and then an optional a.

**Solution:** `a*(c|bb)c(cc)*a?`

Describe the language (with an english sentence) represented by: a+b?c+b?d*

**Solution:** at least one a, followed by an optional b, then at least one c, then an optional b, and then any number of d’s.
Regex to DFA Examples

- Draw a DFA for the language [ab]*
- Draw a DFA for the language [ac]*bd+a?
Draw a DFA for the language [ab]*

Solution:

Regex to DFA Example Solution 1
Regex to DFA Example Solution 2

Draw a DFA for the language \([ac]^*bd+a\)?

Solution:

```
Regex to DFA Example Solution 2

Draw a DFA for the language [ac]^*bd+a?

Solution:
```

```
Diagram:
```

```
```
Regex to DFA (and DFA to Regex) Practice

- Draw a DFA for the language $a^*(cc)^*$
- Draw a DFA for the language $(bb|ac)c^*[ad]$
- Write a regex that matches the same language accepted by the following DFA:
Draw a DFA for the language $a^*(cc)^*$

**Solution:**

![DFA Diagram]

- $q_0$ is the start state.
- $q_1$ is a final state.
- $q_2$ is another final state.
- Transitions:
  - $a$: from $q_0$ to $q_1$.
  - $c$: from $q_0$ to $q_2$, from $q_1$ to $q_2$. 

Regex to DFA Practice Solution 1
Draw a DFA for the language \((bb|ac)c^*[ad]\)

Solution:
Write a regex that matches the same language accepted by the following DFA:

Solution: \((aa|b)b(dd)^*c\)
CFG - Context Free Grammar

- **Context Free**
  - The meaning of something is independent of what’s around it

- **CFGs are more powerful than regular languages**
  - CFGs can do everything regex/DFA/NFA can do and more!
  - CFGs *can* do bracket matching (parenthesis checking), regex *cannot*!
Parse Tree Example

Consider the following grammar

\[
\begin{align*}
\langle E \rangle & \rightarrow \langle E \rangle + \langle T \rangle \\
& \rightarrow \langle E \rangle - \langle T \rangle \\
& \rightarrow \langle T \rangle \\
\langle T \rangle & \rightarrow \langle T \rangle * \langle F \rangle \\
& \rightarrow \langle T \rangle / \langle F \rangle \\
& \rightarrow \langle F \rangle \\
\langle F \rangle & \rightarrow ( \langle E \rangle ) \\
& \rightarrow \text{int}
\end{align*}
\]

Example: draw the parse tree for \( (3 * (2 + 7)) / 6 \) with this CFG
Parse Tree Example Solution

Draw the parse tree for $(3 \times (2 + 7)) / 6$
with this CFG.
Parse Tree Practice

Consider the following grammar (same as example)

\[
\begin{align*}
\langle E \rangle & \rightarrow \langle E \rangle + \langle T \rangle \\
& \rightarrow \langle E \rangle - \langle T \rangle \\
& \rightarrow \langle T \rangle \\
\langle T \rangle & \rightarrow \langle T \rangle * \langle F \rangle \\
& \rightarrow \langle T \rangle / \langle F \rangle \\
& \rightarrow \langle F \rangle \\
\langle F \rangle & \rightarrow ( \langle E \rangle ) \\
& \rightarrow \text{int}
\end{align*}
\]

Draw the parse trees for: 3 * 4 - (8 + 2 / 4) and 1 + 2 - 3 + (5 / (3 + 3) - 15)
Parse Tree Practice Solution 1

Draw the parse tree for \(3 \times 4 - (8 + 2 / 4)\)
Parse Tree Practice Solution 2

Draw the parse tree for \( 1 + 2 - 3 + \frac{5}{3 + 3} - 15 \)
Assignment reminders

● Submit a .txt XOR a .pdf for each question
  ○ Do not submit both for the same question!
  ○ You may submit a .pdf for one question and a .txt for a different question

● Make sure your diagrams and tables are clear and easy to read
  ○ Make sure to leave enough space