Slide 1.
Hi welcome. My project was on Building a regular expression Interpreter.

Slide 2.
Let me ask you, “If regex is a sequence of characters, why do we need an algorithm to work with them?” During this presentation, I’m hopeful to explain the answer.

Slide 3.
So what exactly was I working on you may be wondering. I was working on a regex parser and an creating an NFA with Thompson’s algorithm.

Slide 4:
More about the project.
Define a regular expression syntax
Operators
Character Set
Operator Precedence
Development of the parser
Tokenize regex input
Recursive descent approach
Development of NFA
Create a data structure to represent the NFA

Slide 5: Why does Regex matter?
Regex provides a clear, hands-on demonstration of how abstract theoretical concepts, translate into real-world tools used in search engines, syntax parsing,

Slide 6: Email validation example
to check if a given string is a valid email address by ensuring it contains an "@" symbol, a domain name, and a top-level domain (like ".com").

Slide 7: Regular expression syntax

Now that you understand why it maters lets dig deeper into my syntax: Operators
Concatenation (AB): Matches A followed by B
Union(A|B): Matches either A or B
Kleene Star (A*): Matches zero or more occurrences of A.
Parentheses ((A|B)*): Group expressions for operator precedence.
Character Set
Single characters: a,b,c, etc.
Operational: Support predefined sets
Operator Precedence

Parentheses (( ))
Kleene Star (*)
Concatenation (AB)
Union (A|B)

Slide 8:Parser Development

Tokenize regex input
Convert the regex string into a sequence of tokens (e.g., a , |, *, (,)).
Recursive descent approach

Slide 9: Thompson's Construction
Algorithm

Before I get into the syntax and operation of the creation of the NFA I want to take some time to explain what this algorithm is.

Slide 10: Simple rules
Method of converting regular expressions to their respective NFA diagrams

5 Simple Rules:
An Empty Expression
A symbol
Union Expression
Concatenation(Intersection) Expression
Kleene Star Expression

Slide 11 - 17: Example: a(a|b)*b

1. Now that you know the rules, let's convert a regular expression a(a|b)*b. So where do we start? At the beginning. The first symbol is an "a", so we follow rule #2, but we come across it twice so we have to add it to the NFA stack twice.
2. Now that you know the rules, let's convert a regular expression a(a|b)*b. So where do we start? At the beginning. The third symbol is an "b", so we follow rule #2 again and add it to the NFA stack.
3. The | symbol is used for the Union expression, which means we need to use rule #3. So what we have to do is pop NFA3, and NFA2 to make a new NFA4.
4. Now we are handling the kleene star operation. So we will take NFA4 and kleene star it to create a new NFA5. By the way everytime we create a new NFA we always have to declassify the old NFA, in this case NFA4, the start state and accept state because we have new start and accepts for NFA5.
5. Following rule #4, concatentation, we start by popping off the two NFA's to combine,

NFA5, NFA1 which creates NFA6.

6. We've done this before so now we are processing the input 'b' following rule #2.
7. Now time for the final rule using rule #4, concatenation. I won't go into that much detail, do the same steps as before and you should have a final NFA. And at this point our project would relaize it doesn't have anymore to read there for pop the only NFA left and that is our Final NFA.

Slide 18: NFA Development
Data structure for representing NFAs
States: A set of states.
Transitions: A dictionary mapping states to possible transitions (character or epsilon).
Start State: The initial state of the NFA.
Accept State: The final state(s) of the NFA.

Now that you have some understanding of thompson's algorithm, lets get into how I developed the NFA. First I created a data structure to represent NFAs…

Slide 19-20: Please Watch my product demo
Display video

Slide 21: Thank you