LEXICAL ANALYSIS

Source code ⟶ Tokens ⟶ Assembly code

Lexical Analysis — Parser (Syntactic Analysis)

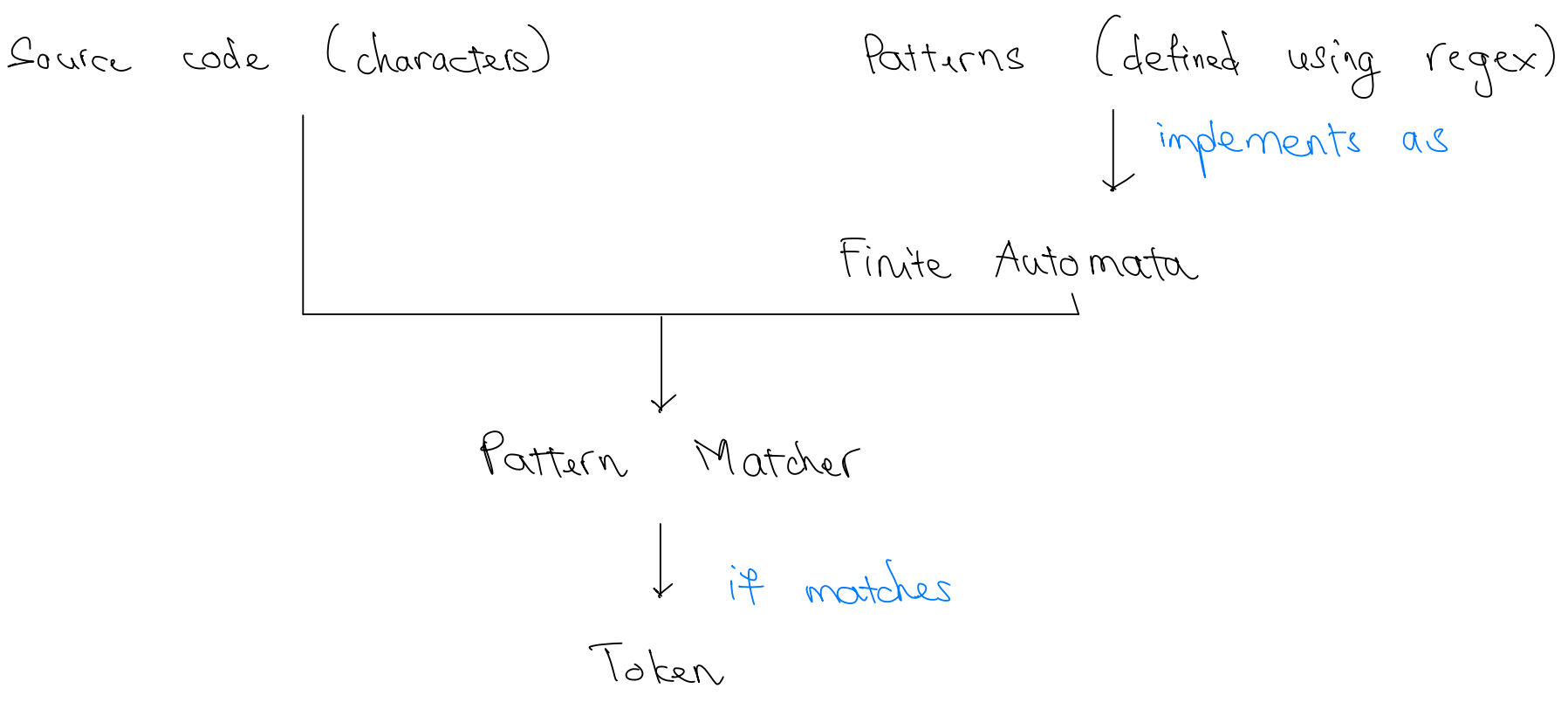
Definition: Lexical Analysis

Lexical Analysis, also known as Tokenizer or Lexing, breaks down source code text into sequence of small units called tokens.

These tokens can be

- keywords (int, String, ...)
- identifiers (this_token, var1, ...)
- literals (10, "hello")
- operators (*, +, ...)

How: Lexical Analyzer turns programs into tokens ?



Example:

```python
# Define token patterns
patterns = [
    ('KEYWORD', r'if|else|while|for'),
    ('IDENTIFIER', r'[a-zA-Z_]\w*'),
    ('NUMBER', r'\d+'),
    ('OPERATOR', r'[+\-*/=]'),
    ('WHITESPACE', r'\s+'),
]

def tokenize(code):
    tokens = []
    position = 0
    while position < len(code):
        match = None
        for token_type, pattern in patterns:
            regex = re.compile(pattern)
            match = regex.match(code, position)
            if match:
                text = match.group(0)
                if token_type != 'WHITESPACE':
                    tokens.append((token_type, text))
                position = match.end()
                break
        if not match:
            print(f"Error: Unexpected character '{code[position]}'")
            position += 1
    return tokens

# Example usage
source_code = "if x = 5 + y"
print(tokenize(source_code))
```