AUTOMATA THEORY

Tutorial 4

TABLE OF CONTENTS

- Assignment updates
- Regex Parsing
- CYK algorithm

ASSIGNMENT UPDATES

- Assignment 2 will be released soon
- An example input file for Programming assignment II (Q3 & Q4) will be uploaded by tonight, the pdf is updated to fix question numbers are running command

PARSING

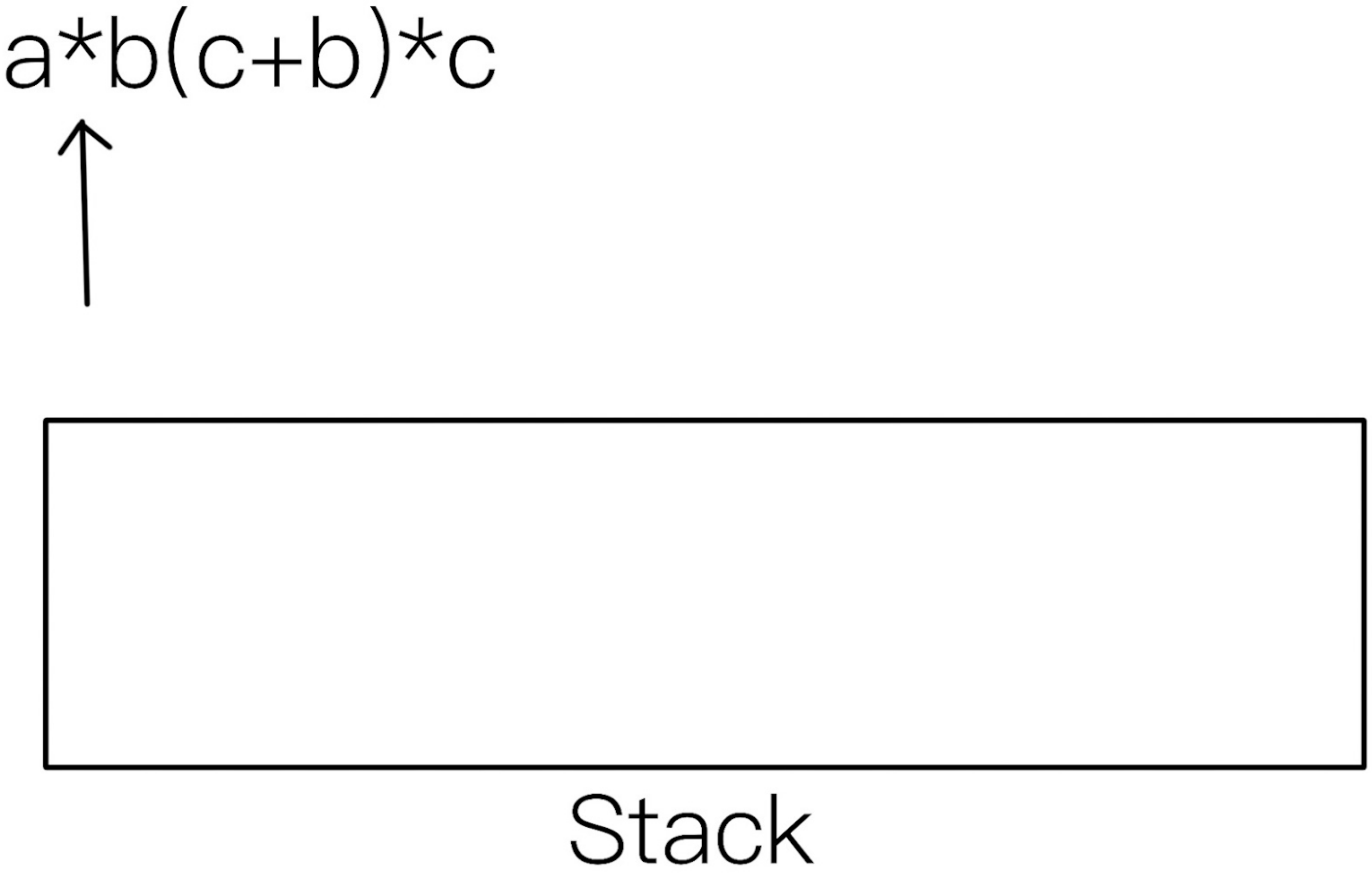
- Automata theory is pervasively used in all branches of informatics to model situations or phenomena classifiable as time & space discrete systems.
- Formal language: A language whose form & meaning are algorithmically defined, I.e. a computer must be able to determine it's correctness and derive it's meaning. All programming languages are formal languages
- Translation from a programming language to machine code is known as compilation, to check if a program will compile is a decision problem. Compilers do more than just checking; they convert the program into meaningful tokens.

PARSING IN PROGRAMMING ASSIGNMENT

- The assignment is to help you understand the conversion from regex to NFA, we do not expect you to learn parsing on your own!
- Regex requires a very simple parsing algorithm which can be implemented using a stack.

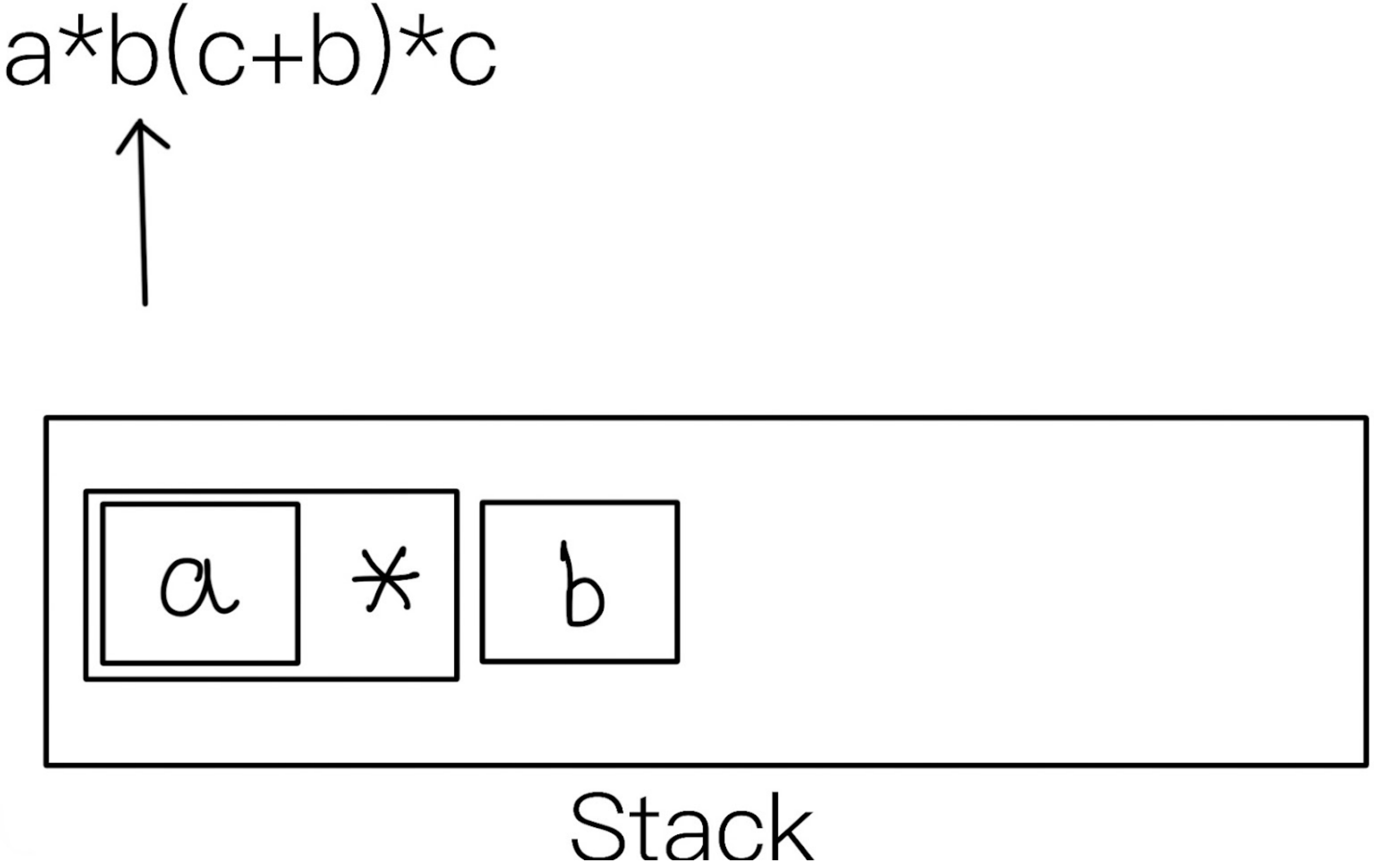
REGEX PARSING

Regex to be parsed: a*b(c+b)*c



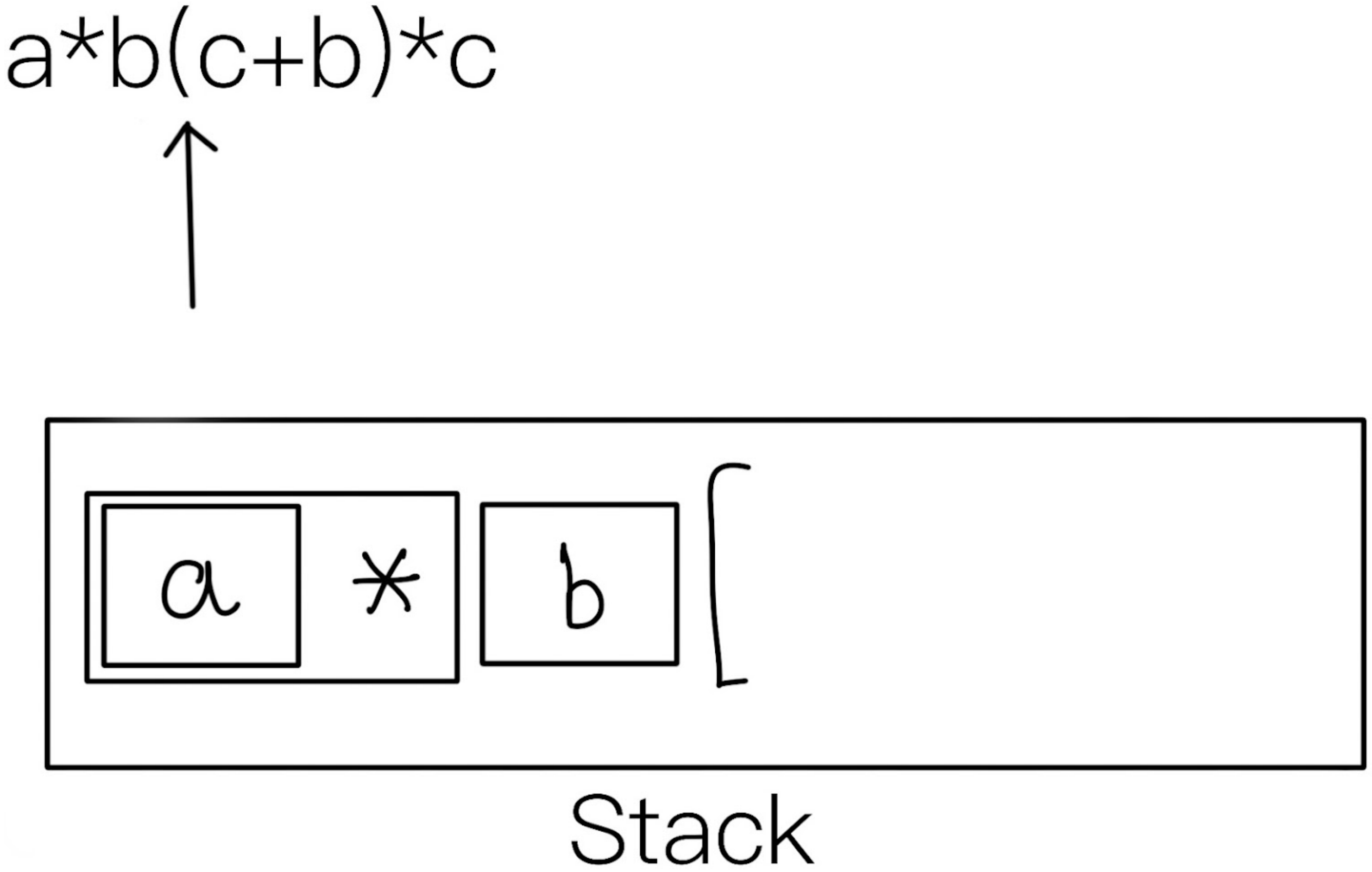
REGEX PARSING

Regex to be parsed: a*b(c+b)*c



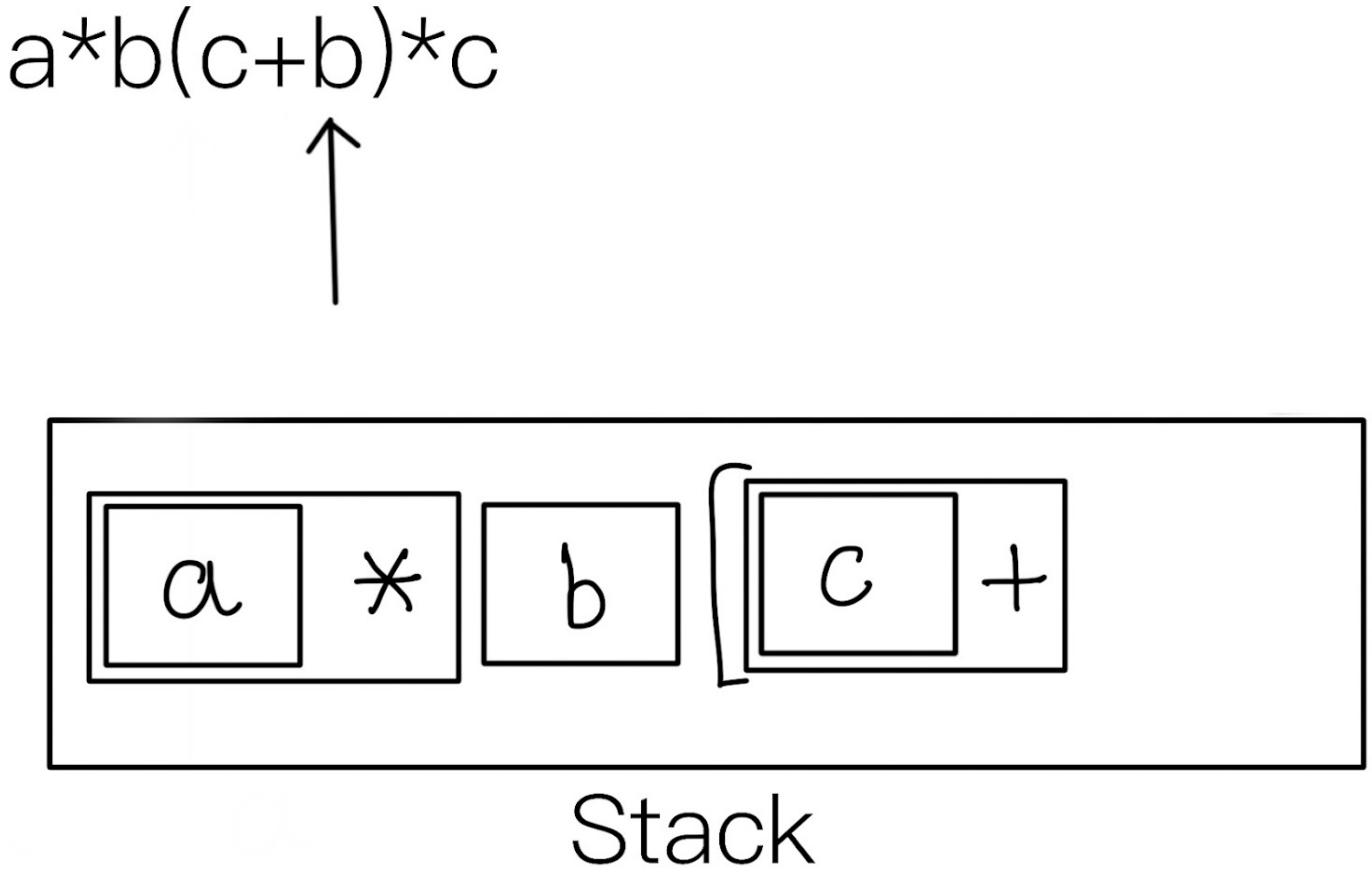
REGEX PARSING

Regex to be parsed: a*b(c+b)*c



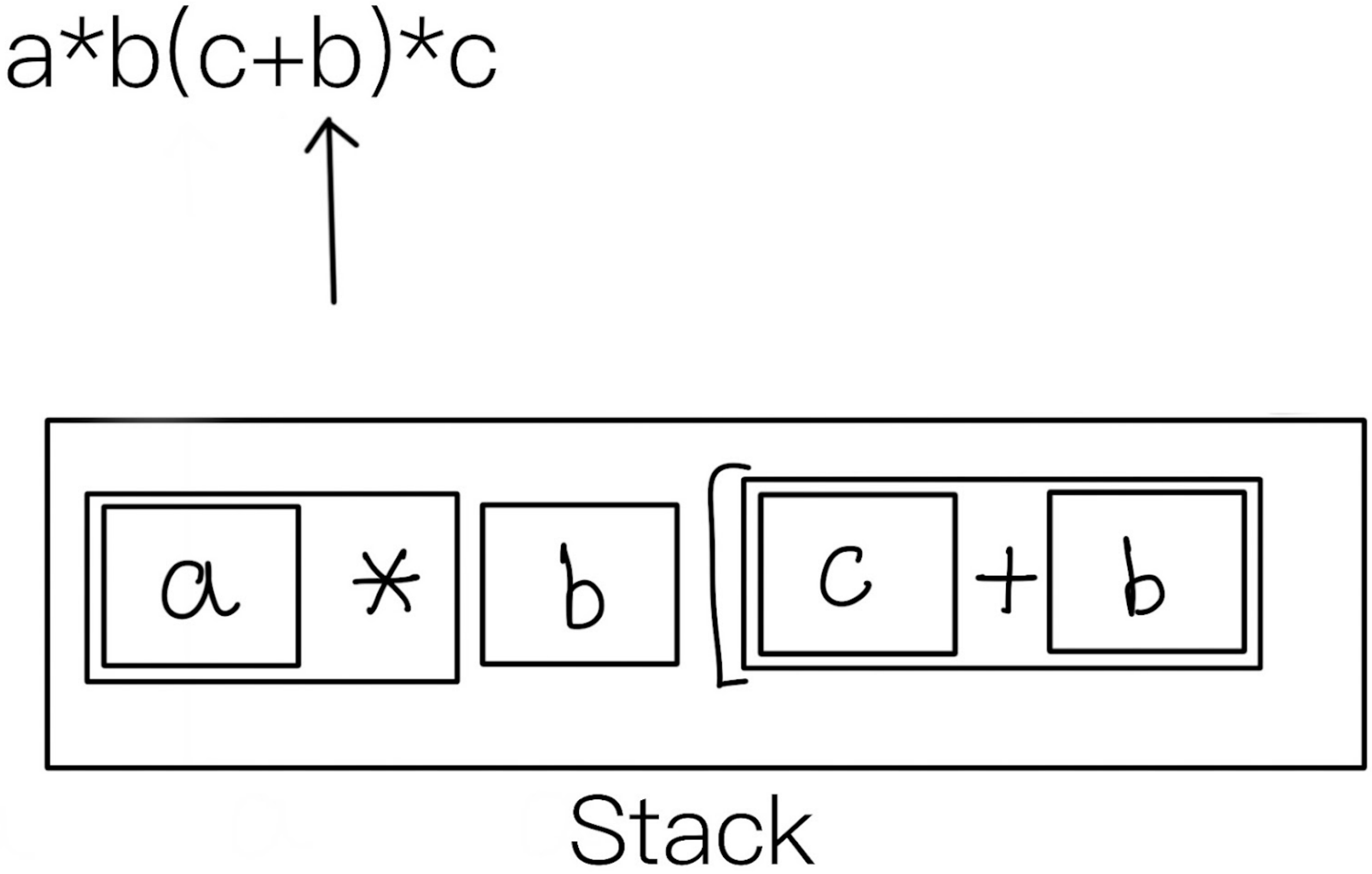
REGEX PARSING

Regex to be parsed: a*b(c+b)*c



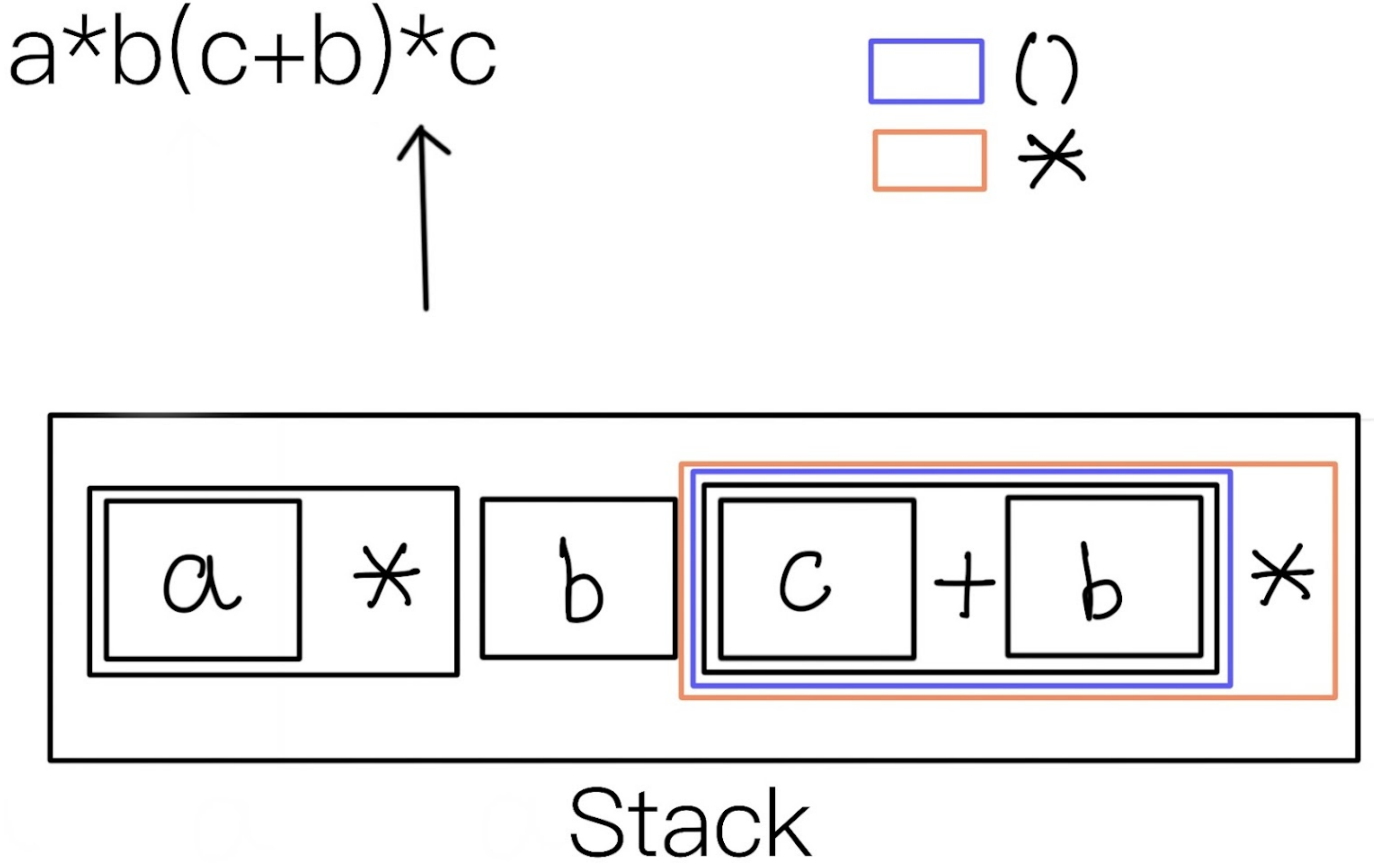
REGEX PARSING

Regex to be parsed: a*b(c+b)*c



REGEX PARSING

Regex to be parsed: a*b(c+b)*c



QUESTION TO PONDER

Can you model the Above as a PDA?

MODELING THE TOKEN IN C++

- A token can be modelled simply as a struct in C++

```
struct Token {
  //...
  vector<Token> tokens;
};
```

CFG & MEMBERSHIP

Above we were able to parse and validate a regular expression. If a language is more complex, such as a context free one, can we decide the language? What will be the algorithm for deciding the language?

THE MEMBERSHIP PROBLEM

- Given a context-free grammar G and a string w
  - G = (V, ∑ ,P , S) where
    - V finite set of variables
    - ∑ (the alphabet) finite set of terminal symbols
    - P finite set of rules
    - S start symbol (distinguished element of V)
    - V and ∑ are assumed to be disjoint
  - G is used to generate the string of a language
- Question:
  - Is w in L(G)?
  - Algorithm (*CYK*) is a good example of dynamic programming or table filling and runs in time $O(n^3)$, where $n = |w|$.

TESTING MEMBERSHIP AND THE CYK ALGORITHM

- CYK algorithm is a bottom-up dynamic programming parser
- CYK algorithm was independently developed by
  - J. Cocke
  - D. Younger
  - T. Kasami
- The CYK algorithm was developed to solve the membership problem.

CONDITIONS OF THE CYK ALGORITHM

1. Assume G is in CNF or convert the given grammar to CNF.
2. Any CFG can be converted to a weakly equivalent grammar in CNF.
3. Each rule in the grammar must satisfy
   - $A \longrightarrow BC$ at most 2 symbols on right side
   - $A \longrightarrow a$, or terminal symbol
   - $S \longrightarrow \varepsilon$ null string
4. $w = \varepsilon$ is a special case, solved by testing if the start symbol is nullable.

CYK ALGORITHM

- Let w = $a_1 \ldots a_n$.
- We construct an n-by-n triangular array of sets of variables.
- X_{ij} = {variables A | A =>* $a_i \ldots a_j$}. It means A ∈ X_{ij} if the sequence of letters starting from i to j can be generated from the non-terminal A.
- Induction on j-i+1 (length of the substring).
  - Till we reach the length of the derived string w.
- Finally, ask if S is in X_{1n}.

$X_{1,5}$				
$X_{1,4}$	$X_{2,5}$			
$X_{1,3}$	$X_{2,4}$	$X_{3,5}$		
$X_{1,2}$	$X_{2,3}$	$X_{3,4}$	$X_{4,5}$	
$X_{1,1}$	$X_{2,2}$	$X_{3,3}$	$X_{4,4}$	$X_{5,5}$

Table for string 'w' that has length 5

BASIS OR INITIAL CONDITION

- Xii = {A | A -> a_i is a production}.
- For example:
  - CNF of a grammar G
    - S ⟶ AB | BC
    - A ⟶ BA | a
    - B ⟶ CC | b
    - C ⟶ AB | a
  - w is baaba

{B}	{A, C}	{A, C}	{B}	{A, C}
b	a	a	b	a

INDUCTION

- X_{ij} = {A | there is a production A -> BC and an integer k, with $i \le k < j$, such that B is in X_{ik} and C is in $X_{k+1,j}$}.
- $X_{1, 2} = (X_{i, i}, X_{i+1, j}) = (X_{1, 1}, X_{2, 2})$
- {B}{A,C} = {BA, BC}
- Steps:
  - Look for production rules to generate BA or BC
  - There are two: S and A
  - $X_{1, 2}$ = {S, A}

{S, A}				
{B}	{A, C}	{A, C}	{B}	{A, C}
b	a	a	b	a

FINAL TABLE

- Is baaba in L(G)? Yes
    - We can see the S in the set X_{1n} where 'n' = 5
    - We can see the table the cell X_{15} = {S, A, C} then
    - if S $\in$ X_{15} then baaba $\in$ L(G)
- The running time of the algorithm is $O(n^3)$.
- Let r =|N| symbols (the size of the non-terminal set).
- Time complexity - $O(r^2n^3) = O(n^3)$.
- Space complexity: A three dimensions table at size n*n*r or a n*n table with lists of size up to r. So, space complexity - $O(rn^2) = O(n^2)$.

{S, A, C}				
Ø or {}	{S, A, C}			
Ø or {}	{B}	{B}		
{S, A}	{B}	{S, C}	{S, A}	
{B}	{A, C}	{A, C}	{B}	{A, C}
b	a	a	b	a

REFERENCES

- S. C. Reghizzi, L. Breveflieri, A. Morzenti, Formal Languages and compilation, Second Edition, Springer
- J. E. Hopcroft, R. Motwani, J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Second Edition, Addison Wesley, 2001
- T.A. Sudkamp, An Introduction to the Theory of Computer Science Languages and Machines, Third Edition, Addison Wesley, 2006