RAMAIAH UNIVERSITY OF APPLIED SCIENCES

ASSIGNMENT

Course Code	19CSC211A
Course Name	Formal Languages and Automata Theory
Programme	B-Tech
Department	Computer Science and Engineering
Faculty	Faculty of Engineering and Technology

Name of the Student	Subhendu Maji
Reg. No	18ETCS002121
Semester/Year	4th semester /2nd year
Course Leader/s	Prakash P

:

Declaration Sheet				
Student Name	Subhendu Maji			
Reg. No	18ETCS002121			
Programme	B. Tech	Semester/Year	4th semester /2nd year	
Course Code	CSC211A			
Course Title	FORMAL LANGUAGES AND AUTOMATA THEORY			
Course Date		to		
Course Leader	Prakash P			

Declaration

The assignment submitted herewith is a result of my own investigations and that I have conformed to the guidelines against plagiarism as laid out in the Student Handbook. All sections of the text and results, which have been obtained from other sources, are fully referenced. I understand that cheating and plagiarism constitute a breach of University regulations and will be dealt with accordingly.

Signature of the Student		Date	May 20, 2020
Submission date stamp (by Examination & Assessment Section)			

Signature of the Course Leader and date	Signature of the Reviewer and date

Contents

Marking Scheme

Declaration Sheet			
Student Name	Subhendu Maji		
Reg. No	18ETCS002121		
Programme	B. Tech	Semester/Year	4th semester /2nd year
Course Code	CSC211A		
Course Title	FORMAL LANGUAGES AND AUTOMATA THEORY		
Course Date		to	
Course Leader	Prakash P		

Assignment			
Name of Student	Subhendu Maji	Register No.	18ETCS002121

Sections		Marking Scheme	Max Marks	Marks First Examiner	Marks Second Examiner
	A1.1	**Introduction**	01		
	A1.2	**Design and Validation**	08		
	A1.3	**Conclusion**	01		
		Total Assignment Marks	**10**		

Question No. 1

Solution to Question No. 1:

A1.1 Introduction

The question asked above represent a problem that is significant to complier design called syntax analysis, as the name suggests the challenge is to analyze the syntax and see is it fraud or not. This is where we use Context free Grammar (CFG)

- A CFG is a set of rules that define a language.
- CFG's define a formal language.
- Formal languages work strictly under the defined rules and their sentences are not influenced by the context.

I. Basic rules of Grammars

- **Terminals** – These make up the actual context of the final sentence. These can include words or letters depending on which of these is used as the basic building block of a sentence.
  In our case, we will use letters defining our building blocks of our language. So, our terminals will include "0,1, s, y, h, v, l, n, m, f, t, a,3,4" each letter consist of an expansion that's defined later in the solution.
- **Non- Terminals** – these are also called variables. These act as a sub language within the language defined by the grammar. Non terminals are placeholders for the terminals. We can use non terminals to generate different pattern of terminal symbols.
  In our case we will use these non-terminals to generate initiate like "U, L, H, C, B, A, Q, T" each will be emphasized more in later usage.

- **Start Symbol-** a start symbol is a special non terminal that represents the initial string that will be generated by the grammar.
  Now that we know the Start Symbol for this context is "U".

A context-free grammar (CFG) consists of a set of productions that you use to replace a variable by a string of variables and terminals.
The language of a grammar is the set of strings it generates. A language is context-free if there is a CFG for it. Context-free grammars (CFGs) are used to describe context-free languages. A context-free grammar is a set of recursive rules used to generate patterns of strings. A context-free grammar can describe all regular languages and more, but they cannot describe all possible languages. Based on the characteristics of CFG containing recursive rules. **The idea to reach the goal is to add a memory device to an FSA. An FSA with a memory device is called a pushdown automaton (PDA).**

PDA is an automaton with finite states and the memory can be unbounded. With the application of a PDA, it will be able to recognize a CFG. Pushdown Automata is a finite automaton with extra memory called stack which helps Pushdown automata to recognize Context Free Languages. A Pushdown Automata (PDA) can be defined as in a given state, PDA will read input symbol and stack symbol (top of the stack) and move to a new state and change the symbol of stack.

Now in the introduction to the problem defined. Online transaction platforms are vulnerable to frauds in this case the fraud is detected when amount to be sent or received is above the threshold accessed. So to provide security we ask for two kinds of user pin one is login pin to login into the platform and another for the transaction to happen after request of send or receive which is called the transaction pin so if there is any mistake happened there then the online platform directly aborts are moves back. There is one more important place where fraud happening with the details of the previous transaction where more money than the threshold accessed is requested to send or receive then the automata is designed to ask a security question which is provided with a valid answer. So, the user should give a valid answer for his request to be proceed or for the transaction to happen. If not, the automaton is designed in a manner where it aborts and transaction has been cancelled which is considered as a fraud happening.

The specification for the automaton it starts with the login where the login pin is verified followed by transaction which selects a bank after which request for send or receive money is done. If the amount is less than threshold limit then the app asks for transaction pin if its right transaction happens or abort. If the amount is more than threshold limit then the platform asks a security question for a valid answer. So, if the answer is valid complete transaction or abort. These are the ways in which online transaction platforms are secured from fraud happening in these areas.

A1.2 Design and Validation

Specification of Context-Free Grammar (CFG) for the above problem, design of Push Down Automata (PDA) that recognizes sentences from the CFG

Non-Terminals (Variables)	Terminals
$S - Start$	$c - Correct\ Pin$
$L - Login$	$s - Send\ Money$
$B - Bank$	$r - Receive\ Money$
$T - Types\ of\ Transation$	$l - Amount\ less\ than\ threshold$
$A - Choose\ Amount$	$m - Amount\ more\ than\ threshold$
$P - Transaction\ Pin$	$x - Answer\ to\ Security\ Question$
$Q - Security\ Question$	$p - Correct\ Transaction\ Pin$
$F - Transation\ Finished$	$a - Bank\ A$
	$b - Bank\ B$

Let G be the context free grammar for the problem

$$\boldsymbol{G} = (\boldsymbol{V}, \Sigma, \boldsymbol{P}, \boldsymbol{S})$$

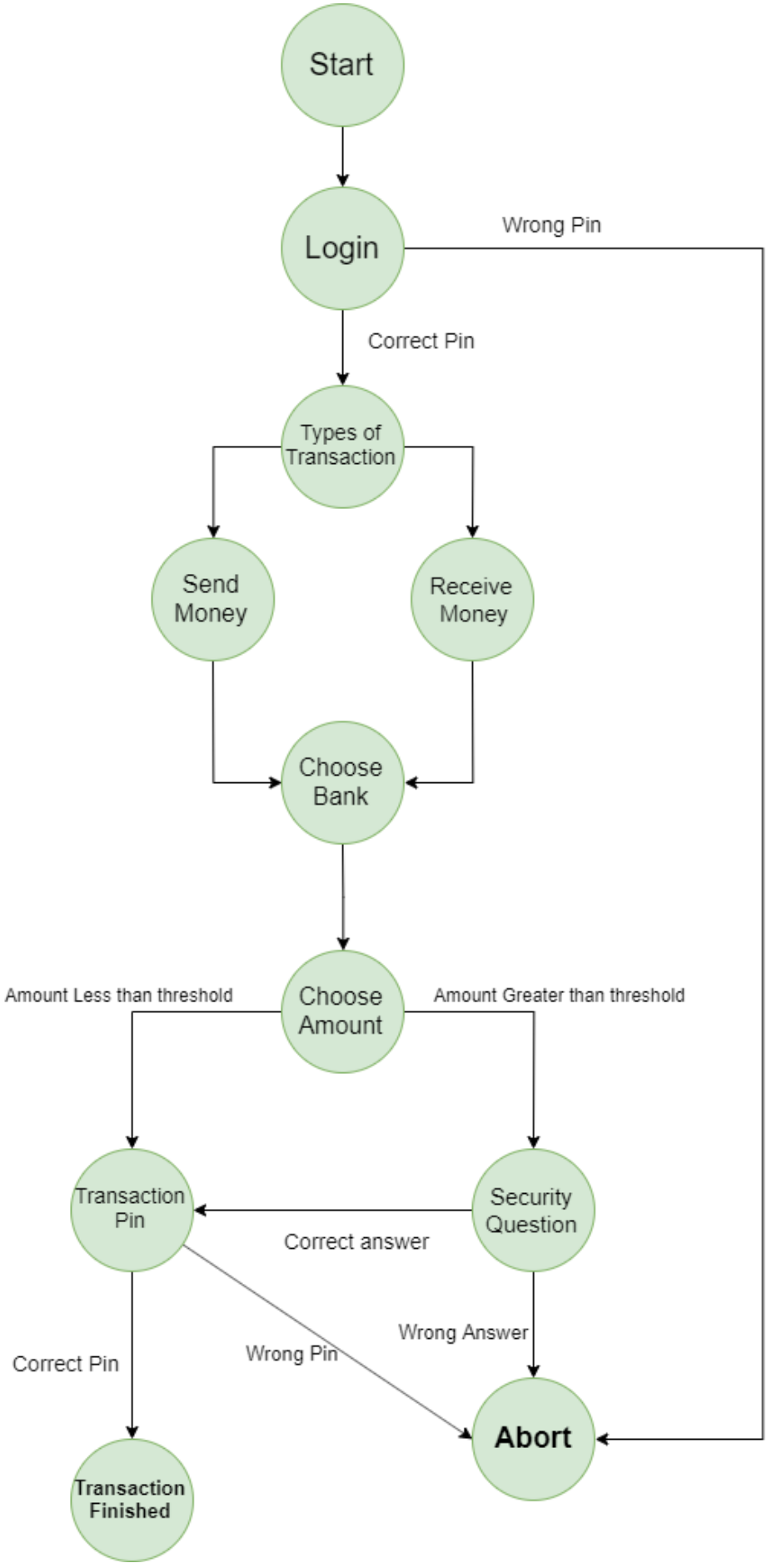
Where, $\boldsymbol{V}$ is the set of Variables, Σ is the set of terminals, $\boldsymbol{P}$ is the Production rule, $\boldsymbol{S}$ is the start symbol

$$\boldsymbol{V} = \{\boldsymbol{S}, \boldsymbol{L}, \boldsymbol{B}, \boldsymbol{T}, \boldsymbol{A}, \boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{F}\}$$

$$\Sigma = \{\boldsymbol{c}, \boldsymbol{s}, \boldsymbol{r}, \boldsymbol{l}, \boldsymbol{m}, \boldsymbol{x}, \boldsymbol{p}, \boldsymbol{a}, \boldsymbol{b}\}$$

Where the productions are,

- $S \rightarrow L$
- $L \rightarrow cT$
- $T \rightarrow sB \mid rB$
- $B \rightarrow aA \mid bA$
- $A \rightarrow lP \mid mQ$
- $Q \rightarrow xP$
- $P \rightarrow p$


Start
Login
Wrong Pin
Correct Pin
Types of Transaction
Send Money
Receive Money
Choose Bank
Choose Amount
Amount Less than threshold
Amount Greater than threshold
Transaction Pin
Security Question
Correct answer
Wrong Answer
Wrong Pin
Correct Pin
Abort
Transaction Finished

Let the equivalent PDA be **P**,

$$\boldsymbol{P} = (\boldsymbol{Q}, \Sigma, \boldsymbol{S}, \boldsymbol{\delta}, \boldsymbol{qo}, \boldsymbol{F}, \boldsymbol{I})$$

Where, $\boldsymbol{Q}$ is the set of states, Σ is the set of input alphabet, $\boldsymbol{S}$ is the set of stack symbols, $\boldsymbol{\delta}$ is the transition function, $\boldsymbol{qo}$ is the initial state, $\boldsymbol{F}$ is the set of accepting symbols, $\boldsymbol{I}$ is the initial stack top symbol

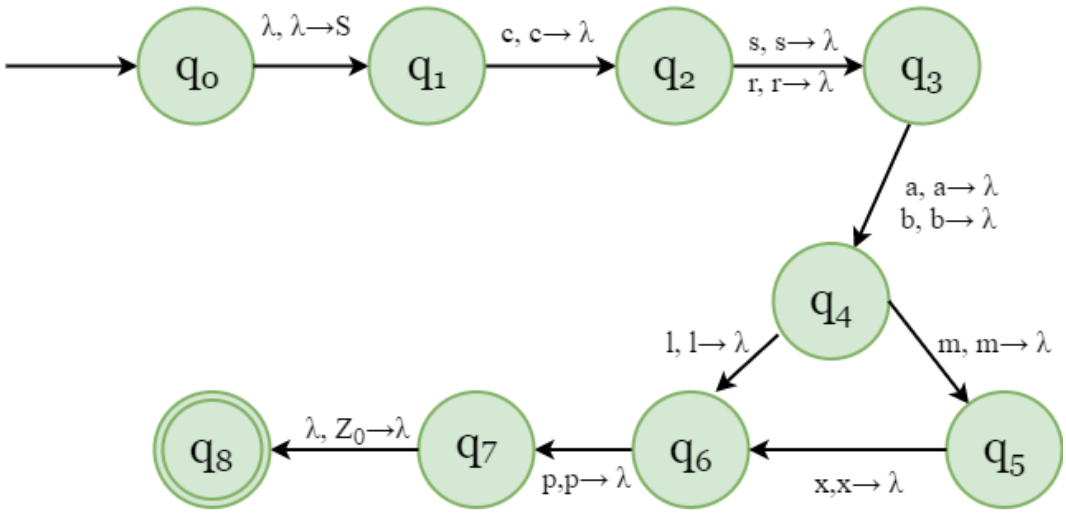
Where $\Sigma = \{c, s, r, l, m, x, p, a, b\}$

, $S = \{c, s, r, l, m, x, p, a, b, S, L, B, T, A, P, Q, F\}$

, $\boldsymbol{I} = z_0$

, And δ –

CURRENT STATE	NEXT STATE
$\delta(q, \lambda, \lambda)$	(q, S)
$\delta(q, \lambda, S)$	(q, L)
$\delta(q, \lambda, L)$	(q, cT)
$\delta(q, \lambda, T)$	$\{(q, sB), (q, rB)\}$
$\delta(q, \lambda, B)$	$\{(q, aA), (q, bA)\}$
$\delta(q, \lambda, A)$	$\{(q, lP), (q, mQ)\}$
$\delta(q, \lambda, Q)$	(q, xP)
$\delta(q, \lambda, P)$	(q, p)
$\delta(q, c, c)$	(q, λ)
$\delta(q, s, s)$	(q, λ)
$\delta(q, r, r)$	(q, λ)
$\delta(q, l, l)$	(q, λ)
$\delta(q, m, m)$	(q, λ)
$\delta(q, x, x)$	(q, λ)
$\delta(q, p, p)$	(q, λ)
$\delta(q, a, a)$	(q, λ)
$\delta(q, b, b)$	(q, λ)

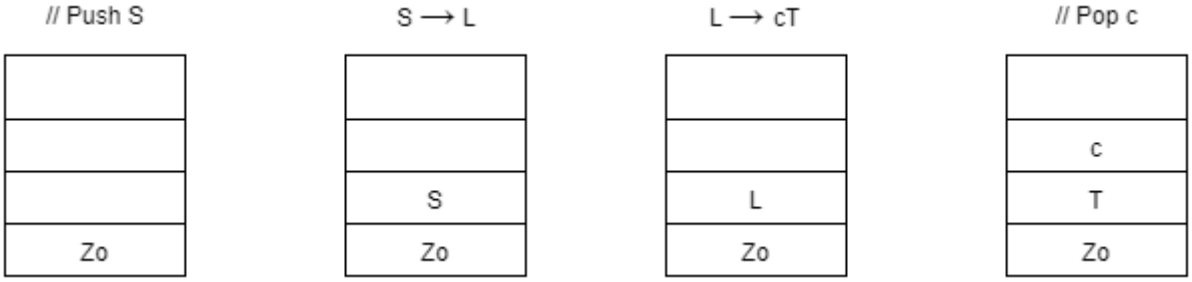


Validation (using stack):

Input string: ***csamxp***

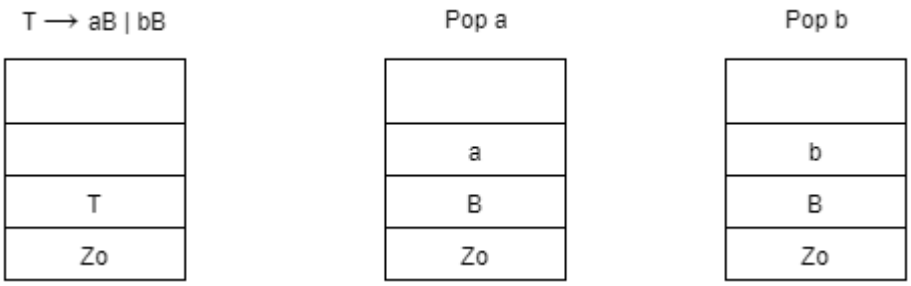
Initially the stack has z_0, to check if we are at the end of the stack, by reading $\boldsymbol{\lambda}$ from the input string, we push start symbol **S** to the top of the stack [$\delta(q, \lambda, \lambda) = (q, S)$]. **S** is replaced by $\boldsymbol{L}$ [$\delta(q, \lambda, S) = (q, L)$]. **L** is replaced by $\boldsymbol{cT}$ [$\delta(q, \lambda, L) = (q, cT)$], now to top of the stack is **c** which is a terminal, so we read from the input string

As the input is $\boldsymbol{c}$, we pop $\boldsymbol{c}$ from the stack. If the input wasn't $\boldsymbol{c}$ then we would abort

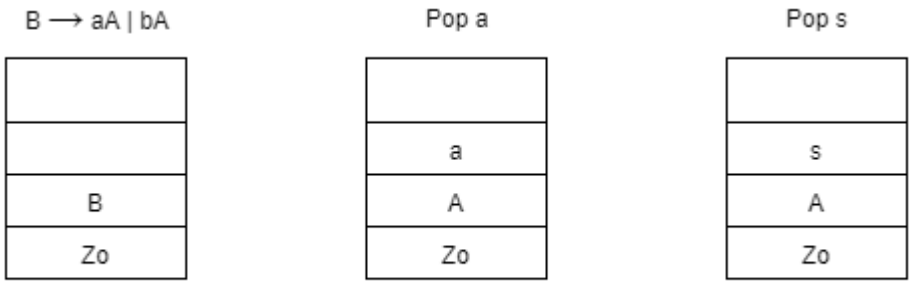


Now, T is at the stop of the stack. $\boldsymbol{T}$ is replaced $\boldsymbol{sB}$ | $\boldsymbol{rB}$ **,** top of the stack is $\boldsymbol{s}$ or $\boldsymbol{r}$ so the we read the input.

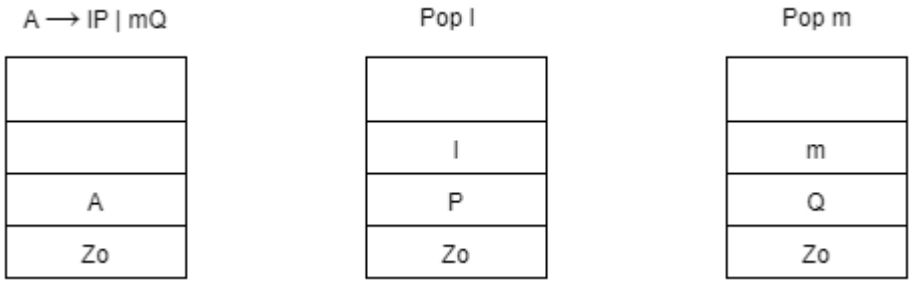
As the input is s, we pop a from the stack



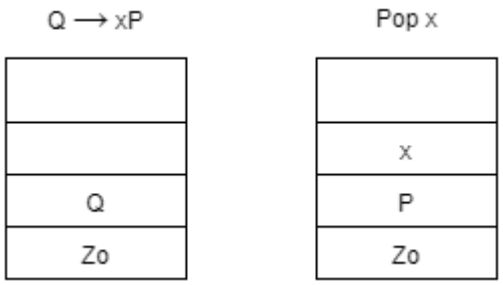
Now the top of the stack is B.
B is replaced by $\boldsymbol{aA \mid bA}$ **,** so we read the input. As the input is $\boldsymbol{a}$, we pop $\boldsymbol{a}$ from the stack.



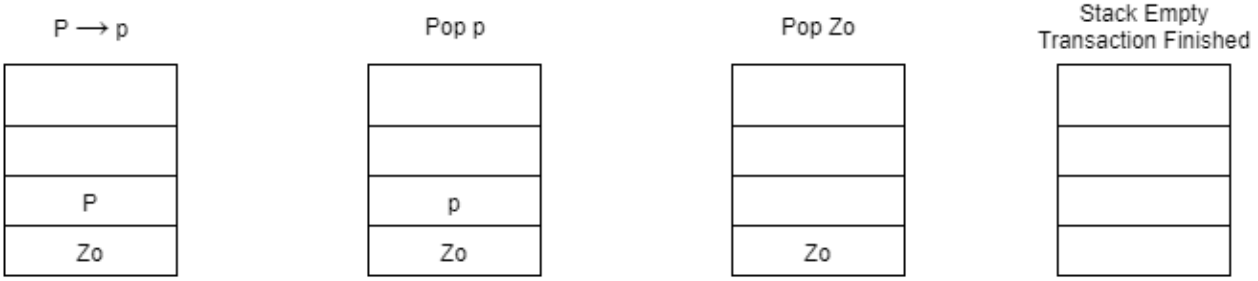
Now A is at the top of the stack. A is replaced by $lP \mid gQ$, we read from the input
As the input is g, we pop g from the stack.



Now Q is at the top of the stack. Q is replaced by xP, we read from the input.
As the input is x, we pop x from the stack.



Now, P is at the top of the stack. P is replaced by p, we read from the input. As the input is p, we pop p from the stack. $z0$ is at the top of the stack, we have reached the end of the stack. zo is popped from the stack; the stack is empty Transaction is Finished.



A1.3 Conclusion

Fraud can be determined in three states:

- By entering the login pin incorrectly that directly aborts.
- By entering the transaction pin incorrectly that automatically aborts.
- By answering the security questions incorrectly, you can determine it as fraud.

✓ The process is NPDA as we can give two or more states as input that's why it's called non deterministic PDA. No specific state is mentioned.

In conclusion, this is a NPDA meaning A non-deterministic pushdown automaton (NPDA), or just pushdown automaton (PDA) is a variation on the idea of a non-deterministic finite automaton (NDFA). Unlike an NDFA, a PDA is associated with a stack (hence the name pushdown). The transition function must also take into account the "state" of the stack.

A PDA is non-deterministic if in some state there are several possible transitions. It doesn't matter if that applies to a transition to a final state. From state $q0$ with $Z0$ on the stack, on reading a there is one possibility. In the same case there is no alternative on input.

So, this online transaction software or application is a non-deterministic pushdown automaton the way it is been develop becomes to come to end there have more than one end state. To develop a platform without fraud we use the NPDA in this application for security where it can abort if there is a problem during transaction.

Bibliography

- https://www.tutorialspoint.com/automata_theory/pda_context_free_grammar.htm
- https://www.geeksforgeeks.org/construct-pushdown-automata-given-languages/
- https://en.wikipedia.org/wiki/Pushdown_automaton