CMSC 621 - Advanced Operating Systems

Project 1

Submitted by:
Aditi Choksi (EW18455)
Akriti Anand (EP52093)

OBJECTIVES:

The objective of this assignment is to provide REST APIs to process the input tokens and to verify them against a PushDown Automata. The program has the following APIs:

- /pdas - returns id and name of all pdas
- /pdas/id - creates a new PDA from the request body provided in the input
- /pdas/id/reset - returns the PDA i.e. the stack and the queue of unprocessed tokens.
- /pdas/id/tokens/position - Process the token for the specified location. The token is provided as a part of the request body. If the token position is not one to be immediately processed, then the token is queued for later processing
- /pdas/id/eos/position - Specifies the position of the last token for the input.
- /pdas/id/is_accepted - Specifies if the PDA is in accepting state or not.
- /pdas/id/stack/top/k - returns the top k symbols from the stack
- /pdas/id/stack/len - return the length of the stack
- /pdas/id/state - returns id and name of all pdas
- /pdas/id/tokens - returns the list of tokens that have been queued for later processing
- /pdas/id/snapshot/k - returns the current state, the queued tokens and the top k symbols in the stack.
- /pdas/id/close - Cleans up the resources generated while processing the input for the pda. If no resources were created, returns nothing to clean.
- /pdas/id/delete - deletes the pda for the given id

How to start the server?

This is a Golang program that is run from the command line.

Server startup Command: `bash start_server.sh`

How to run the clients?

We have 2 different clients creating 2 PDA's of their own

Client 1: `bash client1_script.sh`

Expected Output:

- This bash script runs for the PDA 0^n1^n

- This client demonstrates how correct **tokens presented in random** order can be processed and **accepted**.
- The PDA stores the tokens in a **Hold Back queue** for processing it later and processes it immediately when it can be consumed
- It creates a separate go routine on the server for processing

Client 2: `bash client2_script.sh`

Expected Output:

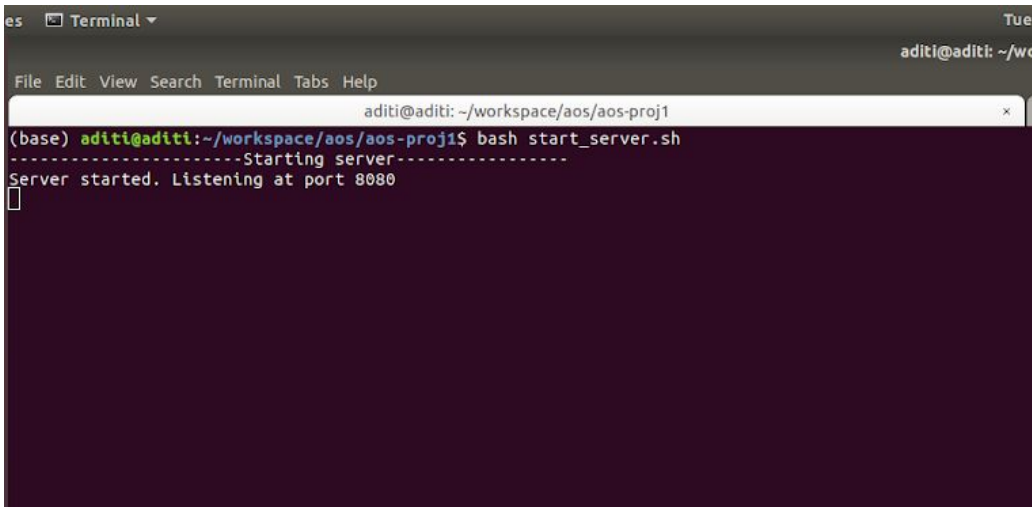
- This bash script runs for the PDA 0^n1^n
- This client demonstrates how **incorrect** tokens presented in **random** order can lead to the PDA **rejecting** the stream of input tokens.
- The PDA stores the tokens in a Hold Back queue for processing it later and processes it immediately when it can be consumed. This
- It creates a thread of its own on the server for processing

PROGRAM ANALYSIS:

- As a part of this assignment we were learnt to create REST APIs in golang. This was very beneficial as REST is the most common form of client server communication used today.
- We have also learnt and implemented multithreading in golang with the help of goroutines.
- Other concepts that we learnt and used as a part of this assignment include - use of go cache, maps and gorilla mux.

OUTPUT SCREENSHOTS:

Start the Server:



Run Test Cases for client 1

```
aditi@aditi: ~/workspace/aos/aos-proj1
File  Edit  View  Search  Terminal  Tabs  Help
aditi@aditi: ~/workspace/aos/aos-proj1    aditi@aditi: ~/workspace/aos/aos-proj1
(base) aditi@aditi:~/workspace/aos/aos-proj1$ bash client1_script.sh

---------------- Create PDA with id 100 -----------------
 PDA created[{"Id":"100","Name":"0n1n"}]

------------Put tokens ---------------
"Token kept in hold_back_Queue"

"Token kept in hold_back_Queue"

---------------- Current state of the PDA ------------------
"q2"

---------------- Current length of stack -------------------
1

-------------- Continue processing other tokens ----------
"Token kept in hold_back_Queue"

"Token kept in hold_back_Queue"

"Token kept in hold_back_Queue"

----------------- Queued Tokens --------------------------
[{"Hold_back_Position":"5","Hold_back_Token":"1"},{"Hold_back_Position":"4","Hold_back_T
oken":"1"},{"Hold_back_Position":"3","Hold_back_Token":"1"},{"Hold_back_Position":"2","H
old_back_Token":"0"},{"Hold_back_Position":"1","Hold_back_Token":"0"}]

----------------------Put token at position 0 ----------------
"Token processed successfully. Please enter the next token"

---------------- Snapshot --------------------------------
{"Topk":["$"],"Current_State":"q3","Hold_back_Queue":[]}

---------------- Call eos --------------------------------

---------------- Call is_accepted() -------------------
"Input tokens successfully Accepted"
(base) aditi@aditi:~/workspace/aos/aos-proj1$
```

Run Test Cases for client 2

```
aditi@aditi: ~/workspace/aos/aos-proj1
(base) aditi@aditi:~/workspace/aos/aos-proj1$ bash client2_script.sh

---------------- Create PDA with id 101 -----------------
 PDA created[{"Id":"100","Name":"0n1n"},{"Id":"101","Name":"0n1n"}]

------------Put tokens ---------------
"Token processed successfully. Please enter the next token"

"Token processed successfully. Please enter the next token"

---------------- Current state of the PDA -----------------
"q2"

---------------- Current length of stack -------------------
3

-------------- Continue processing other tokens ----------
"Token processed successfully. Please enter the next token"

"Input tokens Rejected by the PDA"

---------------- Snapshot ---------------------------------
{"Topk":["$","0"],"Current_State":"q3","Hold_back_Queue":[]}

---------------- Call eos ---------------------------------

---------------- Call is_accepted() -------------------
"Input tokens Rejected by the PDA"

---------------- Reset ---------------------------------

---------------- Snapshot ---------------------------------
{"Topk":[],"Current_State":"q1","Hold_back_Queue":[]}

---------------- Show all Pdas -------------------
[{"Id":"100","Name":"0n1n"},{"Id":"101","Name":"0n1n"}]

---------------- Delete Pdas -------------------
"Pda deleted."

---------------- Show all Pdas -------------------
[{"Id":"100","Name":"0n1n"}]
(base) aditi@aditi:~/workspace/aos/aos-proj1$
```