Push Down Automata That Flip Their Stacks
Presented By,
Bhargavi Belathur Ramalingaiah

Topics to be covered
- Introduction
- Push Down Automata
- Formal Model/Definition
- A Hierarchy of Languages
- Conclusion
Introduction

- A pushdown automata is a finite automata equipped with an auxiliary storage device in the form of stack.
- An additional power to the automata by allowing it to flip stacks, it allows the automata to read and write at both ends of the stack-double ended queue.
- De-queue can simulate two stacks and hence the computational model becomes equivalent to the Turing machine model.

Push Down Automata

- It is like a NFA but has an extra component called stack.
- Stack has an unlimited memory and restricted access.
- Access to the stack, for both reading and writing, done only at the top (LIFO)
PDA $P$ with $\{WW^R : W \in \{0,1\}^*\}$, as per class note

Formal Model /Definition of Flip stack PDA:
The formal computational model is a flip stack pushdown automata (FPDA) defined as follows,

- The FPDA is defined by a six-tuple $PDA = (Q, \Sigma, \Gamma, \delta, q_0, F)$
  
  Where $Q$ is a finite set of states,
  
  $\Sigma$ is a finite input alphabet,
  
  $\Gamma$ is a finite stack alphabet,
  
  $\delta$ is the transition function (A Stack-flip function - $\Delta$)
  
  $q_0$ is the initial state,
  
  $F$ is the set of final states.

Such that - FPDA $F = (P, \Delta)$
The word can be pushed or popped from both the ends when the stack is flipped.

The language can be accepted by FPDA

\[ L = \{ww : w \in \{0,1\}^*\} \] is not a CFL.

Suppose, \( W = "100" \) and the stack is flipped once then it becomes double ended queue where we can push or pop from both the ends hence we can pop the word = “100” in a same order hence we can obtain the language “WW” through Flip stack pushdown automata.
A Hierarchy of Languages

- Using FPDA, a hierarchy of languages is defined for $k \geq 0$,
  where $k$ = number of flips allowed:
  - $L(k) = \{ L : L = L(F(k)), \text{for some FPDA } F \}$ and
    for $k$ unlimited number of flips:
  - $L(\infty) = \{ L : L = L(F), \text{for some FPDA } F \}$

Proposition 3.1

- Prove the following:
  1. $L(0) \equiv \text{CFL}$
  2. $L(\infty) \equiv \text{All recursively enumerable languages}$

- 1. It is clear that a stack that cannot be flipped is equivalent to the stack in a PDA.
- 2. Proving that the FPDA with infinite flips is equivalent to a Turing Machine is equivalent to proving part 2.
Contd..

PDA with 2-stacks, then it can accept any language as Turing machines.

There is a discussion in the textbook of how a Turing Machine is equivalent to a two-stack PDA. Clearly, a two-stack PDA is equivalent to the FPDA $F$, where for each movement of $P$, the machine $F$ will require one stack reversal.

- When all the inputs symbols are pushed onto stack $S_1$, once the last symbol of $w$ has been read.
- Pop $S_1$ and push the symbols one by one to $S_2$ (next state)
- Once it's done, move onto next state and Pop $S_2$
- If all inputs are exhausted, input string is accepted.
Turing Machines

- Turing machine can simulate k-flips PDA easily.

State control

read-write head

A A B - - - ...

infinite tape

TM has an infinite-input tape for both read write the elements wherein 2-stack PDA is just a special case of a 3-tape TM that uses it's tapes restricted way.

TM is more flexible and accurate model, which can do everything as a real computer do.

Theorem 3.2

The language class (0) is properly contained in the language class (1).

Show that \( L \in L(1) \). In other words, that it can be described by a PDA that flips its stack once.

- When the stack is flipped once, given string of the form \( ww \), the FPDA accepts it as proved in earlier slide. Thus \( L \in L(1) \).
Contd..

- L(1) proves that $L=\{ww:w \in \{0,1\}^*\}$ accepted FPDA.
- CFL = $L(0) \subseteq L(1) \subseteq L(2) \subseteq \ldots \subseteq L(\infty) = \text{r.e.}$ (recursively enumerable)
- It’s known that CFL=L(0) And also $L(k) \subseteq L(k+1)$ wherein Languages which are in L(k+1) could not be in L(k), but L(k+1) consists of L(k) and also defines extra language.

How do we show that each language class in the hierarchy is non-empty?!

For each $k$, define a language $L_k$ as follows:

- $L_0 = \{x_1x_1r : x_1 \in \{0,1\}^*\}$, as proved earlier
- $L_1 = \{x_1x_1 : x_1 \in \{0,1\}^*\}$, as proved earlier
- $L_2 = \{x_1x_2x_1x_2r : x_1, x_2 \in \{0,1\}^*\}$
- $L_3 = \{x_1x_2x_3x_1x_3x_2r : x_1, x_2, x_3 \in \{0,1\}^*\}$

... 

- $L_{2t} = \{x_1x_2 \ldots x_{2t} x_1x_2tx_2tx_2r2t-1r \ldots xtxt+1r : x \in \{0,1\}^*\}$ (even)
- $L_{2t+1} = \{x_1x_2 \ldots x_{2t+1}x_1x_2t+1rx_2rx_2tr \ldots xt-1x_2txt+1r : x \in \{0,1\}^*\}$ (odd)
Contd..

- Each language class in the hierarchy is non empty as each levels will define the language for their levels.
- Each level will define the previous levels and also with extra sync.
- Suppose, if L(k) defines k then L(k+1) defines L(k) plus additional languages hence each levels in hierarchy is non empty.
Conclusion

- Stack with no flips is equivalent to machines that produce Context Free Languages.
- Stack with infinite flips is equivalent to machines that produce recursively enumerable languages (Turing Machine).
- A hierarchy of languages can be formed where the languages with no flips (L(0)) is contained within languages with one flip, two flips, etc. (L(1), L(2)…)

Reference

1. Pushdown automaton with the ability to flip it’s stack paper by Palash Sarkar: http://eccc.hpi-web.de/report/2001/081/
2. Introduction to Automata Theory, Languages, and Computation by John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman.
3. Class notes
Thank You!