Contents

Pushdown Machines

$\triangledown$ is bottom of a stack

- Input Symbols = $($, $)$, $\dashv$
- Stack Symbols = L, $\triangledown$
- States `{s}`
- Transitions = `(, L, s`
  - Push `(L)`
  - Advance
  - Move to state `(s)`
- Starting stack $\triangledown$

	(	)	$\dashv$	
L	Push `(L)`, advance, state `(s)`	Pop, advance, state `(s)`	Reject	
$\triangledown$	Push `(L)`, advance, state `(s)`	Reject	Accept	State `(s)`

Starting stack $\triangledown$

Stack	Input
$\triangledown$	`(()())`$\dashv$
$\triangledown$`L`	`()())`$\dashv$
$\triangledown$`LL`	`)())`$\dashv$
$\triangledown$`L`	`())`$\dashv$

Accept $\dashv$

Errors

- Unmatched left parenthesis
- Extra right parenthesis

Build a pushdown machine to recognise $1^N 0^N$ where `n>0`

- Input Symbols $= 0, 1, \dashv$
- Stack Symbols $= L, \triangledown$
- States `{s1, s2}`
- Transitions `1, L, s`
  - Push `(L)`, advance
  - Move to state `(s)`

State `(s1)`

	`0`	`1`	$\dashv$
L	Pop, advance, state `(s2)`	Push `(L)`, advance, state `(s1)`	Reject
$\triangledown$		Push `(L)`, advance, state `(s1)`	Reject

State `(s2)`

	`0`	`1`	$\dashv$
L	Pop, advance, state `(s2)`	Reject	Reject
$\triangledown$	Reject	Reject	Accept

Replace

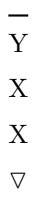
- Another stack operation
- `replace (xyz)`
  - pop, push(x), push(y), push(z)

Y = state where expect to see 1

$$\begin{array}{c} \overline{} \\ \text{Y} \\ \triangledown \\ \overline{} \\ 2 \end{array}$$

`1100`

`100`



`00`

State `(s)`

	0	1	⊣
X	Pop, advance		
Y	Pop, retain	replace(xy), advance	
▽			Accept

Starting stack: $\triangledown Y$

Stack	Input
$\triangledown Y$	1100 ⊣
$\triangledown XY$	100 ⊣
$\triangledown XXY$	00 ⊣
$\triangledown XX$	00 ⊣

Stack	Input
$\triangledown X$	$0 \dashv$
$\triangledown$	$\dashv$

Pushdown Translator

A pushdown translator is simply a pushdown recogniser that produces an output string

> Design a machine to convert a string of `0`s and `1`s into a string of the form 1^N0^M where `N` and `M` are the number of `1`s and `0`s respectively

- $101011 \Rightarrow 11100$
- Push `Z` onto stack when see a `0`
- Output `1` when see a `1`

- Pop `Z`s
- Result: `11100`

	0	1	⊣
Z	Push(Z), advance	Out(1), advance	Out(0), Pop, Retain
▽	Push(Z), advance	Out(1), advance	Accept

Stack	Input	Output
▽	01011 ⊣	
▽Z	1011 ⊣	
▽Z	011 ⊣	`1`
▽ZZ	11 ⊣	`1`

Stack	Input	Output
▽ZZ	1 ⊣	`11`
▽ZZ	⊣	`111`
▽Z	⊣	`1110`
▽	⊣	`11100`