1 Introduction

One of the things that limits the power of finite state automata is that they have no dynamic memory. All the information they contain must be encoded in the current state, and this is a static resource. Our next class of machines, the *push down automata* remedy this by adding a single stack to the control structure. It turns out that this is still quite limiting – the languages accepted by push down automata are all context free. A *pumping lemma* for context free languages, somewhat more complex than that for regular languages establishes that while we can now finally recognize the $a^n b^n$ language, we're still stuck with $a^n b^n c^n$. In a nutshell, the problem is that the stack memory is "use once" and can't be refreshed.

2 Definitions

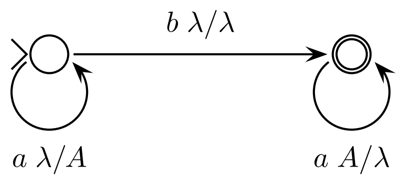
A *push down automaton* is a nondeterministic finite state automaton augmented by a stack. So its parts are:

Q	a finite set of states,
Σ	the input alphabet (lower case letters)
Γ	the stack alphabet (upper case letters)
δ	a transition function
q_0	the initial state
F	the set of accepting states

Some more details about the transition function – it takes as input the current state, an input letter (or λ) and the stack top letter (or λ). It produces as output a set of possibilities (non determinism) each of which consists of a new state and a new stack top (or λ). The interpretation is: consume the input letter (if any), pop the original stack top (if any), move to the new state, and push the new stack top (if any).

A computation proceeds in the normal way following various transitions as allowed by the input (and choosing them arbitrarily if more than one is available). The exact conditions for acceptance are that at the end of the computation no more input should remain, the stack should be empty, and the state should be an accepting one.

Despite their complexity transitions can be represented relatively compactly in diagrams.



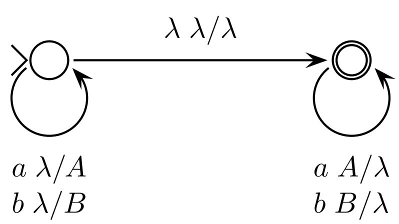
In this machine the three transitions behave as follows:

$a\ \lambda/A$	read a, ignore the stack top, push A
$b\ \lambda/\lambda$	read b, leave the stack alone
$a\ A/\lambda$	read a, pop A from stack top, don't push

To accept, all the pushes that occur in the left state must match pops in the right state, so the language accepted by this push down automaton is $\{a^n b a^n \mid n \geq 0\}$ – certainly a non-regular language.

3 Facts about push down automata

The use of non-determinism is essential for push down automata to gain power. For instance consider the language of even length palindromes over $\{a, b\}$. This is easily accepted by a push down automaton:



But, it's essential that the machine can "guess" when to stop pushing and start popping.

The following result is important, but the proof is technical and uninspiring.

Theorem 3.1. *The languages accepted by push down automata are precisely the languages that have context free grammars.*

4 Pumping lemma for context free languages

For a change, theorem first, then proof as is more traditional.

Theorem 4.1. *Let L be a context free language. There is a positive integer k such that for all $z \in L$ with $|z| \geq k$ we can write $z = uvwxy$ such that:*

$$|vwx| \leq k$$
$$|v| + |x| > 0$$
$$uv^i wx^i y \in L \quad \text{for all } i \geq 0.$$

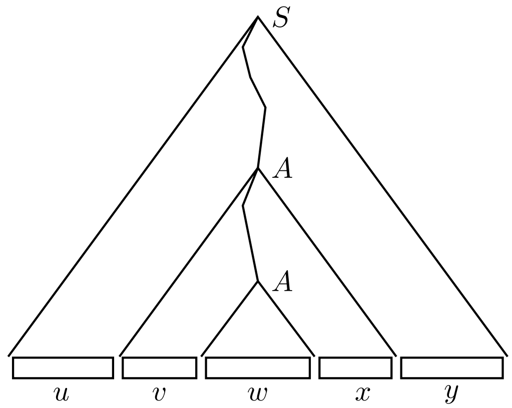
This looks rather like the pumping lemma for regular languages except we need to split the part that is to be pumped into two pieces. The proof will also make use of the "repeated configuration" idea, except in the derivation tree of a word.

Proof. Choose a grammar for L. Each rule is of the form:

$$A \to B_1 B_2 \dots B_t$$

where the B_i are either terminals or non-terminals. Let m be the maximum length of the right hand side of a rule, and let j be the total number of non-terminal letters. Choose $k = m^{j+1}$.

Let $z \in L$ be given with $|z| > k$. Since the maximum possible branching factor in its derivation tree is m, the depth of the tree must be greater than $j + 1$. So, some branch of the tree has at least $j + 1$ internal nodes. Since these are labelled with non-terminals, there is a repeated label on this branch, in fact there is a repeated label among the last $j + 1$ internal nodes of this branch. This gives us the situation pictured below:



Now the theorem follows because we can either replace the top A with the bottom one, eliminating v and x, or the bottom one with a duplicate of the top one which gives us an extra repetition of v and x – and this latter construction can be repeated as often as we like. $\square$

As with the pumping lemma for regular languages, this pumping lemma is used to prove that certain languages are not context free. For example, consider the language:

$$L = \{a^n b^n c^n \mid n \geq 0\}.$$

We will show that this is not context free by contradicting the pumping lemma. So, suppose it were context free and choose k as given by the pumping lemma. Let $w = a^k b^k c^k$ and factor $w = uvwxy$ as guaranteed by the pumping lemma. Since $|vwx| \leq k$ the substring vwx contains at most two out of the three characters a, b, and c. But in that case in uv^2wx^2y at least one character occurs exactly k times (one that doesn't belong to vwx) while some other character occurs more than k times (one that does belong to vwx). Hence, $uv^2wx^2y \notin L$ and we have our desired contradiction. So, L cannot be context free.

5 Closure properties for context free languages

The following result is relatively easy to prove either by the manipulation of grammars or of push down automata.

Theorem 5.1. *Let L_1 and L_2 be context free languages, and let R be a regular language. The following languages are all context free: $L_1 \cup L_2$, $L_1 L_2$, L_1^*, and $L_1 \cap R$.*

However, the intersection of two context free languages is not necessarily context free:

$$\{a^n b^n c^k \mid n \geq 0,\ k \geq 0\} \cap \{a^n b^k c^k \mid n \geq 0,\ k \geq 0\} = \{a^n b^n c^n \mid n \geq 0\}.$$

Therefore, the complement of a context free language need not be context free either (if it were, then because unions are, intersections would be too).

6 Tutorial problems

1. Build a PDA to accept each of the following languages:

   (a) Equal, the set of strings having the same number of a's as b's in any order.

   (b) $\{a^n b^n c^m \mid n, m \geq 0\}$.

   (c) BalancedParentheses, the set of strings over $\{(,), a, b\}$ in which the parentheses are properly balanced (and the other symbols can occur arbitrarily).

   (d) (*) PostFix, the set of strings over $\{a, b, +, -\}$ that represent legitimate expressions written in postfix notation, where $+$ is a binary operator, and $-$ a unary operator.

   Briefly, a and b represent values and $+$ and $-$ represent operators. There is a stack to hold values – any input which is a value is pushed onto the stack. Any input which is an operator, causes either one (in the case of $-$) or two (for $+$) values to be popped off the stack – the operator is applied, and the result is pushed back on to the stack. Since we aren't really computing anything for this exercise, this can be simulated by just modifying the stack size appropriately. An expression is legitimate if there are always enough symbols in the stack for any operator that arrives, and after processing it completely, there is exactly one symbol in the stack.

2. Apply the pumping lemma and write out detailed arguments showing that the following languages are not context-free:

   (a) $\{a^n b^m a^n b^m \mid n, m \geq 0\}$.

   (b) $\{a^p \mid p \text{ is prime}\}$.

   (c) $\{a^n b^n a^n \mid n \geq 0\}$.

3. Show, by intersection with a suitable regular language and deriving a contradiction, that Square, the set of all words of the form ww, where $w \in \{a, b\}^*$ is not context-free. (NB – look up the page ...)