CSCI 340: Computational Models

Pushdown Automata

Chapter 14

Department of Computer Science

A New Format for FAs

- Regular Languages are a strict subset of Context-Free Languages
- We want to make a machine that can *accept* CFGs/CFLs.
- But where to begin? Let's start with an FA.

Step 1: Input String $\rightarrow$ Input Tape

We always had some *input string* for an FA, but it was never formally stored anywhere. Now, let's introduce an *input tape*.

- It must be long enough to support any arbitrary-length string, so it is *infinitely long* — yikes, that's expensive $$$
- Some people use the silly term "half-infinite" for this condition (which is like being half sober)

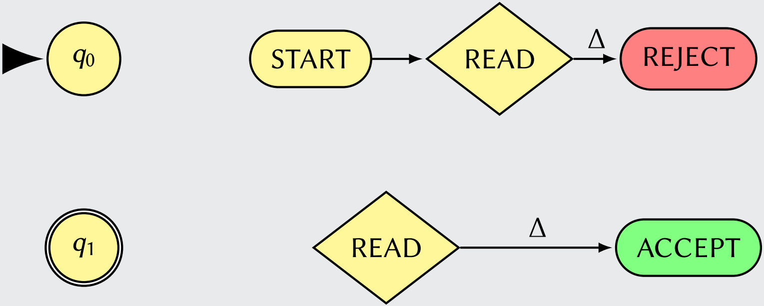
aaab

a	a	a	b	Δ	

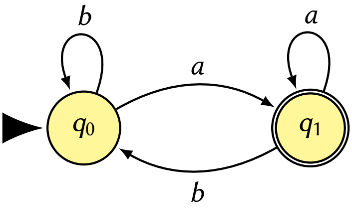
A New Format for FAs

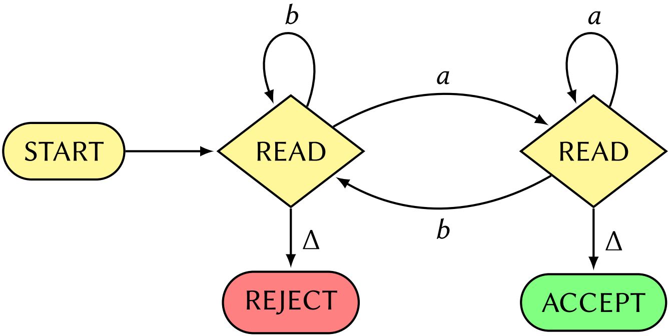
Step 2: Separating Initial/Final/Reject

With FAs, a state could be initial, accepting, both, or neither.
At each state we always *read* a single letter.



A New Format for FAs (Example)



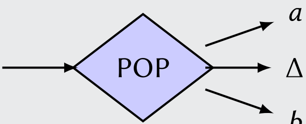


Leaving the Realm of FAs

Step 3: Introduce a Pushdown Stack

- We have introduced these new primitives so we can easily add two additional operations: PUSH α and POP
- We will leverage the use of a stack to *remember* information which we can react on later
- PUSH α will add the character α to our stack. NOTE: what is stored on the stack does not need to match Σ

- POP will remove the "top" character on the stack and *react* on it in some way through outgoing edges. If the stack is empty, Δ (the empty character) is "returned"



The Pushdown Automata

1. An alphabet Σ of input letters
2. An input TAPE (infinite in one direction). Initially the string of input letters is placed on the TAPE at the beginning. The rest of the TAPE is blank (filled with Δs)

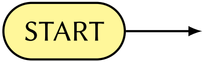
a	a	a	b	a	Δ	Δ	Δ

3. An alphabet Γ of STACK characters
4. A pushdown STACK (infinite in one direction). Initially empty.

Δ
·
·
·

The Pushdown Automata

5. One START state that has only out-edges

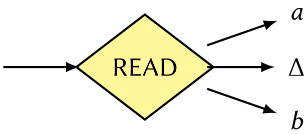


6. Halt states, ACCEPT and REJECT, with only in-edges.

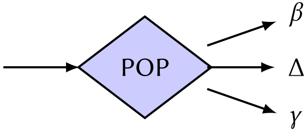
7. Finitely many non-branching PUSH states that introduce characters from Γ onto the stack

The Pushdown Automata

8. Finitely many branching states of two kinds:
    1. States that read the next unused letter from the TAPE. This is a READ state which may have out edges from Σ or Δ. This can be non-deterministic.



    2. States that read the top character of the STACK. This is a POP state which may have out edges from Γ or Δ. This can be non-deterministic.



“Running” Input on a Pushdown Automaton

- To **run** a string of input letters on a PDA means to begin from the START state and follow the unlabeled edges and labeled edges that apply to produce a path through the graph.
- This path will either end at a halt state or crash in a branching state when there is no corresponding edge when read/popped.
- When letters are read from the TAPE or characters are popped from the STACK, they are used up and “vanish”
- An input string with a single path that ends in ACCEPT is said to be **accepted**.
- An input string that can follow a selection of paths is said to be **accepted** IFF at least one of the paths leads to ACCEPT
- The set of all strings accepted by a PDA is called the **language accepted** by the PDA, or the **language recognized** by the PDA

PDAs and Regular Languages

Theorem

For every regular language L, there is some PDA that accepts it

Proof.

Because L is regular, it is accepted by some FA. The constructive algorithm converting an FA to a PDA was shown at the beginning of this presentation. □

Major differences between PDAs and FAs

- The length of the path formed by a given input may be different for PDAs and FAs.
- A string of 7 letters will have an accept or reject path of exactly 7 edges long.
- For a PDA, it may be much shorter or longer (as it depends on the the number of READs, POPs, and PUSHes encountered, or if ACCEPT or REJECT were prematurely encountered)

PDA Reduction

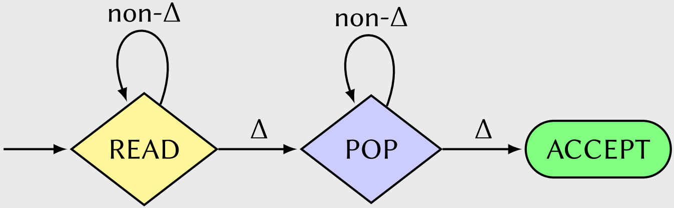
Theorem

Given any PDA, there is another PDA that accepts exactly the same language with the additional property that whenever a path leads to ACCEPT, the STACK and the TAPE contain only blanks (Δ).

Proof by Constructive Algorithm.

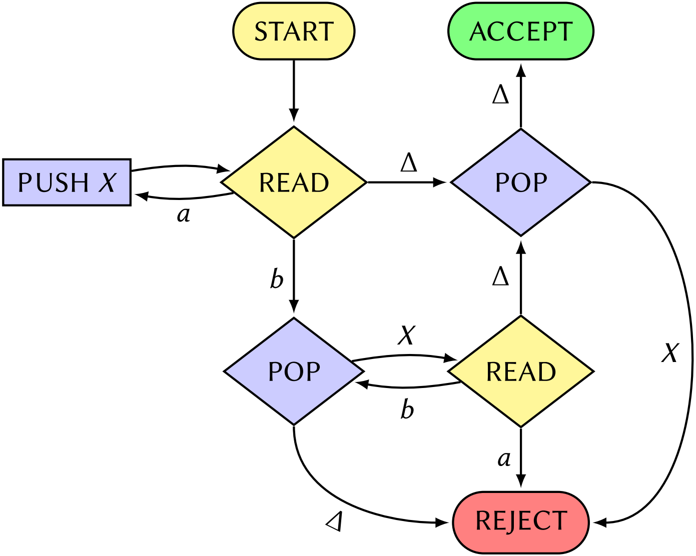
Whenever we have the machine part:

Replace it with:



□

Board Example — Tracing *aaabbb*Δ



Deterministic and Non-Deterministic PDAs

- A **deterministic PDA** is one for which every input string has a unique path through the machine
- A **non-deterministic PDA** is one for which at certain times we may have to choose among possible paths through the machine.
    - If there exists *some* path such that the input string leads to an ACCEPT state, then the input string is accepted
    - If all possible paths lead to REJECT state(s), then the input string is rejected
    - If a choice to be made is not feasible/possible, then the machine **crashes** and the input is rejected
- Non-deterministic PDAs are **more powerful** than Deterministic PDAs (and we will discuss this later)

Example: PALINDROMEX

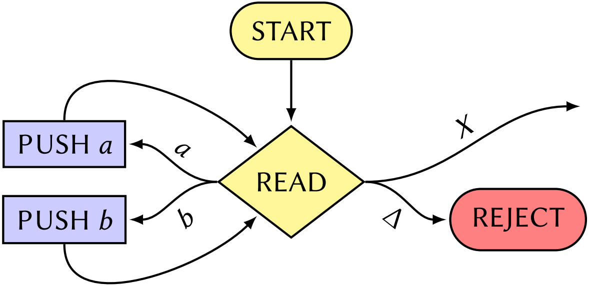
The language PALINDROMEX contains all words of the form:

$$s\ X\ \text{reverse}(s)$$

where s is any string in $(\mathbf{a} + \mathbf{b})^*$. The words in the language are:

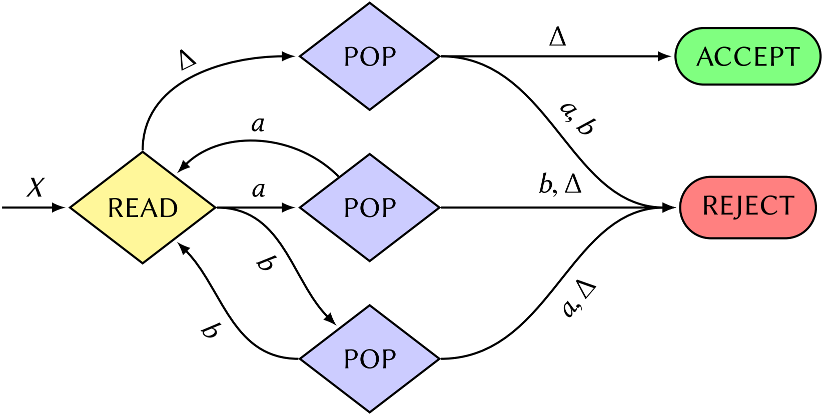
$\{X \;\; aXa \;\; bXb \;\; aaXaa \;\; abXba \;\; baXab \;\; bbXbb \;\; aaaXaaa \;\; aabXbaa \ldots\}$

First part of the machine:



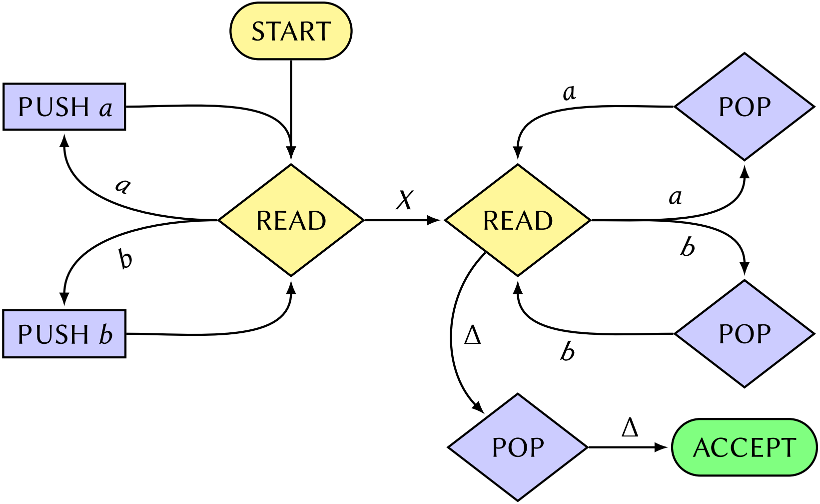
Example: PALINDROMEX

Second part of the machine:



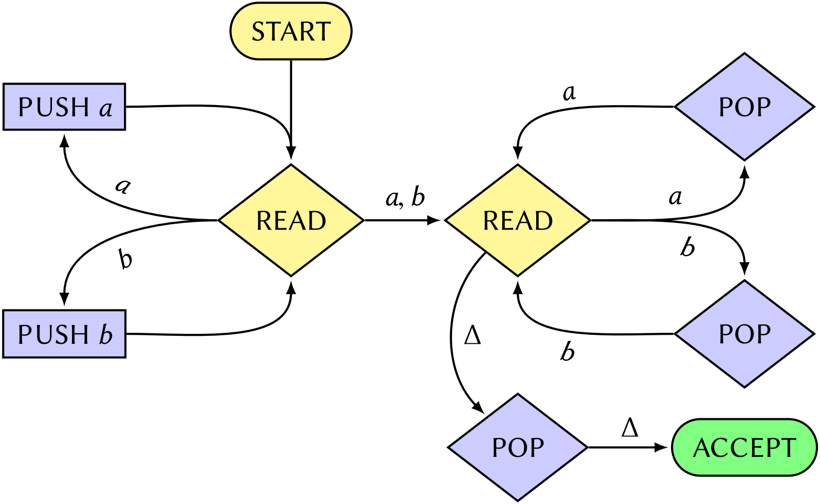
PALINDROMEX — Non-deterministic

Entire Machine (with REJECTs removed)



Example: ODDPALINDROME

Consider a language very similar to PALINDROMEX, but replace X with *a* or *b* — you are left with ODDPALINDROME



Chalkboard Example – EVENPALINDROME

- We have shown that PALINDROMEX and ODDPALINDROME are very similar through PDA construction.
- How different can EVENPALINDROME be?

Homework

For homework problems, consult the course webpage