Chapter 14

Pushdown Automata

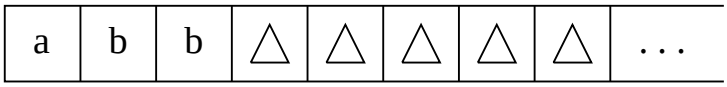
14.1 Introduction

- Previously, we saw connection between
  1. Regular languages
  2. Finite automata
- We saw that certain languages generaged by CFG's could not be accepted by FA's.

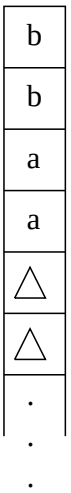
14.2 Pushdown Automata

- Now we will introduce new kind of machine: pushdown automaton (PDA).
- Will see connection between
  1. context-free languages
  2. pushdown automata
- Pushdown automata and FA's share some features, but a PDA can have one extra key feature: STACK.
  - infinitely long INPUT TAPE on which input is written.

- INPUT TAPE is divided into cells, and each cell holds one input letter or a blank Δ.



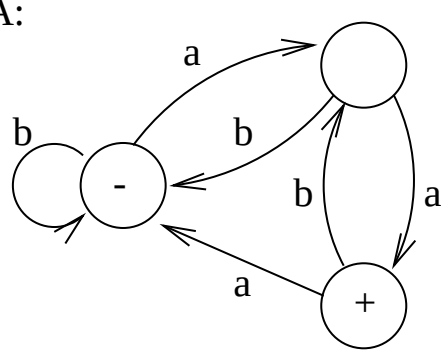
- Once blank Δ is encountered on INPUT TAPE, all of the following cells also contain Δ.
- Read TAPE one cell at a time, from left to right. Cannot go back.
- START, ACCEPT, and REJECT states.
- Once enter either ACCEPT or REJECT state, cannot ever leave.
- READ state to read input letter from INPUT TAPE.
- Also, have an infinitely tall PUSHDOWN STACK, which has last-in-first-out (LIFO) discipline.
- Always start with STACK empty.
- STACK can hold letters of STACK alphabet (which can be same as input alphabet) and blanks Δ.
- Once we encounter a Δ in stack, everything below it is also a Δ.



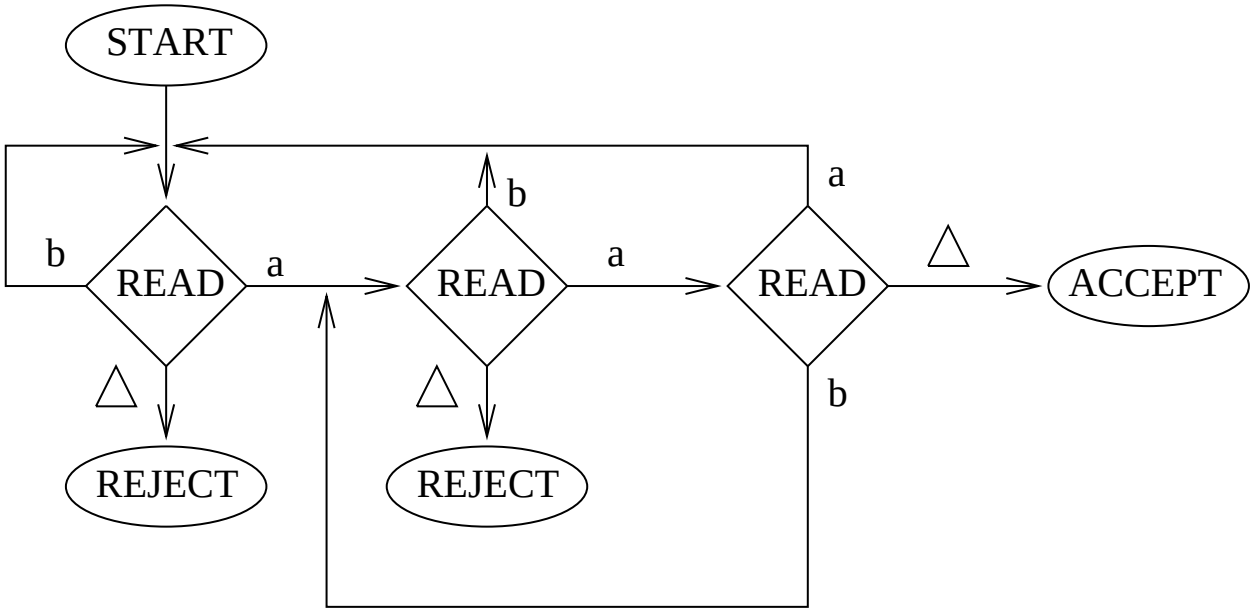
- PUSH and POP states alter contents of STACK.
  - PUSH adds something to the top of the STACK.
  - POP takes off the thing on the top of the STACK.

Example: Convert FA to PDA

FA:



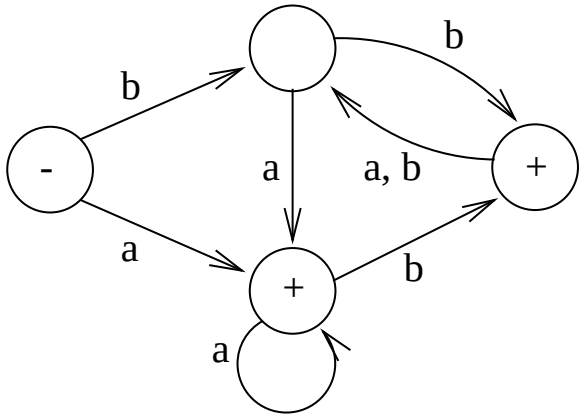
PDA:



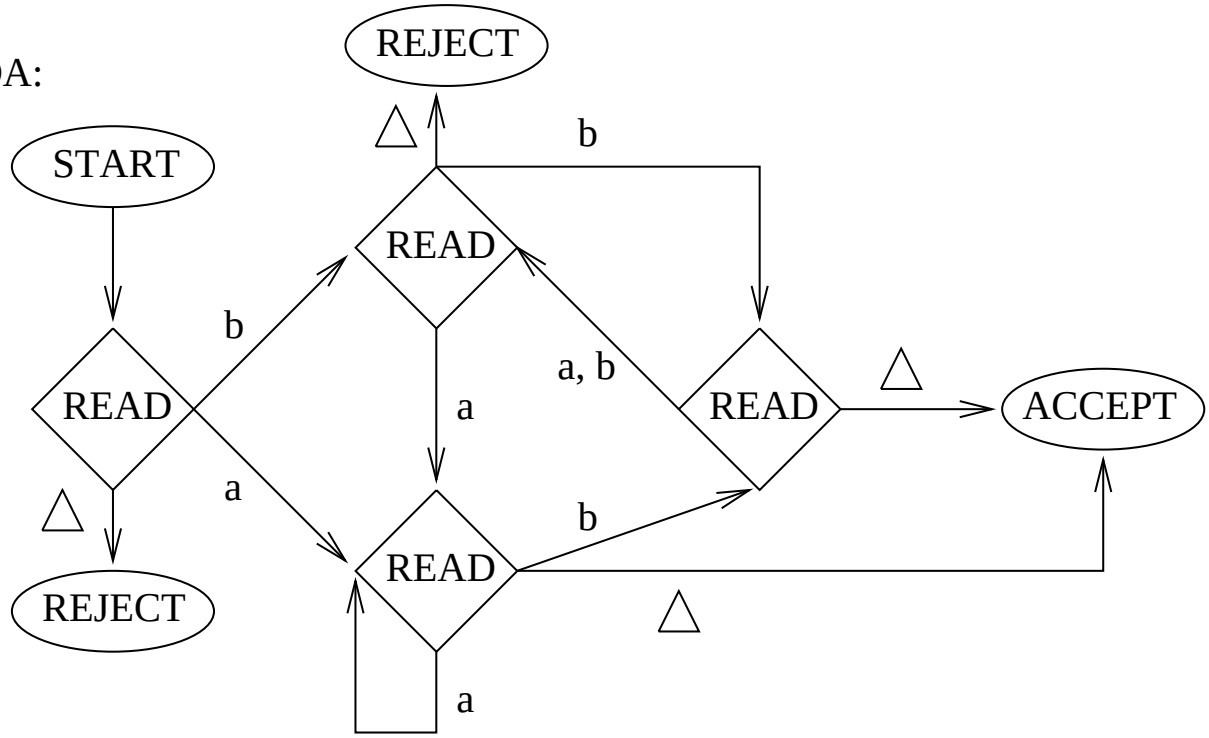
For this example, no STACK.

Example: Convert FA to PDA

FA:

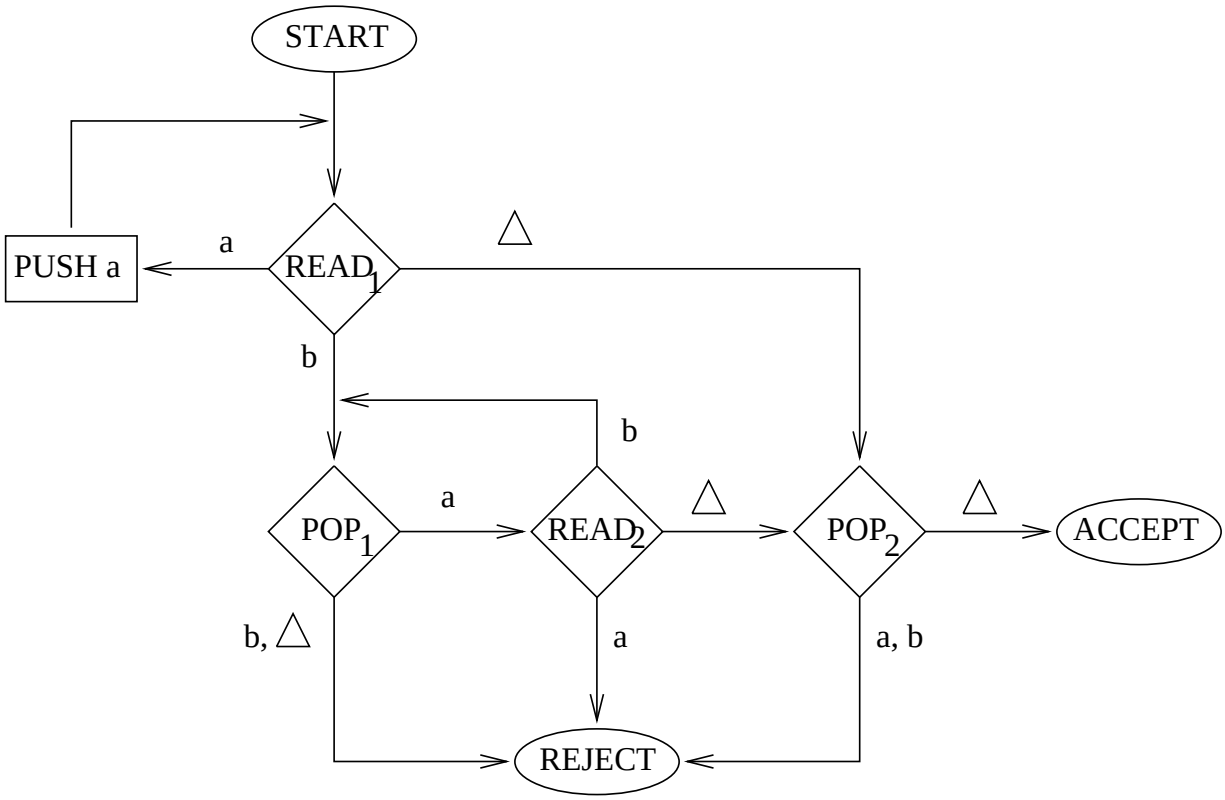


PDA:



For this example, no STACK.

Example: PDA with STACK



Suppose we had the INPUT TAPE

a	a	b	b	Δ	Δ	$\cdots$

Stack is initially empty:

Δ
Δ
$\vdots$

See what happens when we process it:

STATE	STACK	TAPE
START	$\Delta\cdots$	$aabb\Delta\cdots$
READ_1	$\Delta\cdots$	$\not{a}abb\Delta\cdots$
PUSH a	$a\Delta\cdots$	$\not{a}abb\Delta\cdots$
READ_1	$a\Delta\cdots$	$\not{a}\not{a}bb\Delta\cdots$
PUSH a	$aa\Delta\cdots$	$\not{a}\not{a}bb\Delta\cdots$
READ_1	$aa\Delta\cdots$	$\not{a}\not{a}\not{b}b\Delta\cdots$
POP_1	$a\Delta\cdots$	$\not{a}\not{a}\not{b}b\Delta\cdots$
READ_2	$a\Delta\cdots$	$\not{a}\not{a}\not{b}\not{b}\Delta\cdots$
POP_1	$\Delta\cdots$	$\not{a}\not{a}\not{b}\not{b}\Delta\cdots$
READ_2	$\Delta\cdots$	$\not{a}\not{a}\not{b}\not{b}\not{\Delta}\Delta\cdots$
POP_2	$\not{\Delta}\Delta\cdots$	$\not{a}\not{a}\not{b}\not{b}\not{\Delta}\Delta\cdots$
ACCEPT	$\not{\Delta}\Delta\cdots$	$\not{a}\not{a}\not{b}\not{b}\not{\Delta}\Delta\cdots$

The language accepted by the PDA is

$$\{a^n b^n : n = 0, 1, 2, \ldots\}$$

which is a nonregular language.

Proof. see pages 295–299 of text. ∎

So, why can PDA's accept certain nonregular languages?

- STACK is memory with unlimited capacity.
- FA's only had fixed amount of memory built in.

14.3 Determinism and Nondeterminism

Definition: A PDA is *deterministic* if each input string can only be processed by the machine in one way.

Definition: A PDA is *nondeterministic* if there is some string that can be processed by the machine in more than one way.

A nondeterministic PDA

- may have more than one edge with the same label leading out of a certain READ state or POP state.
- may have more than one arc leaving the START state.

Both deterministic and nondeterministic PDAs

- may have no edge with a certain label leading out of a certain READ state or POP state.
- if we are in a READ or POP state and encounter a letter for which there is no out-edge from this state, the PDA crashes.

Remarks:

- For FA's, nondeterminism does not increase power of machines.
- For PDA's, nondeterminism does increase power of machines.

14.4 Examples

Example: Language PALINDROMEX, which consists of all words of the form

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

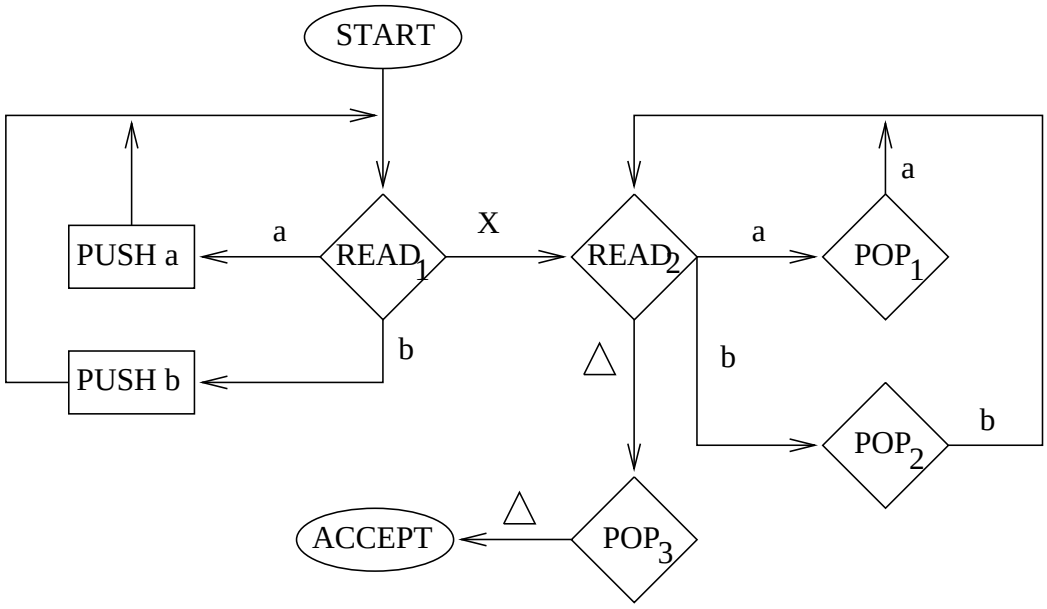
where s is any string generated by $(\mathbf{a}+\mathbf{b})^*$.

$$\text{PALINDROMEX} = \{X, aXa, bXb, aaXaa, abXba, baXab, bbXbb, \ldots\}$$

- Each word in PALINDROMEX has odd length and X in middle.
- When processing word on PDA, first read letters from TAPE and PUSH letters onto STACK until read in X.
- Then POP letters off STACK, and check if they are the same as rest of input string on TAPE.

PDA:

- Input alphabet $\Sigma = \{a, b, X\}$
- Stack alphabet $\Gamma = \{a, b\}$

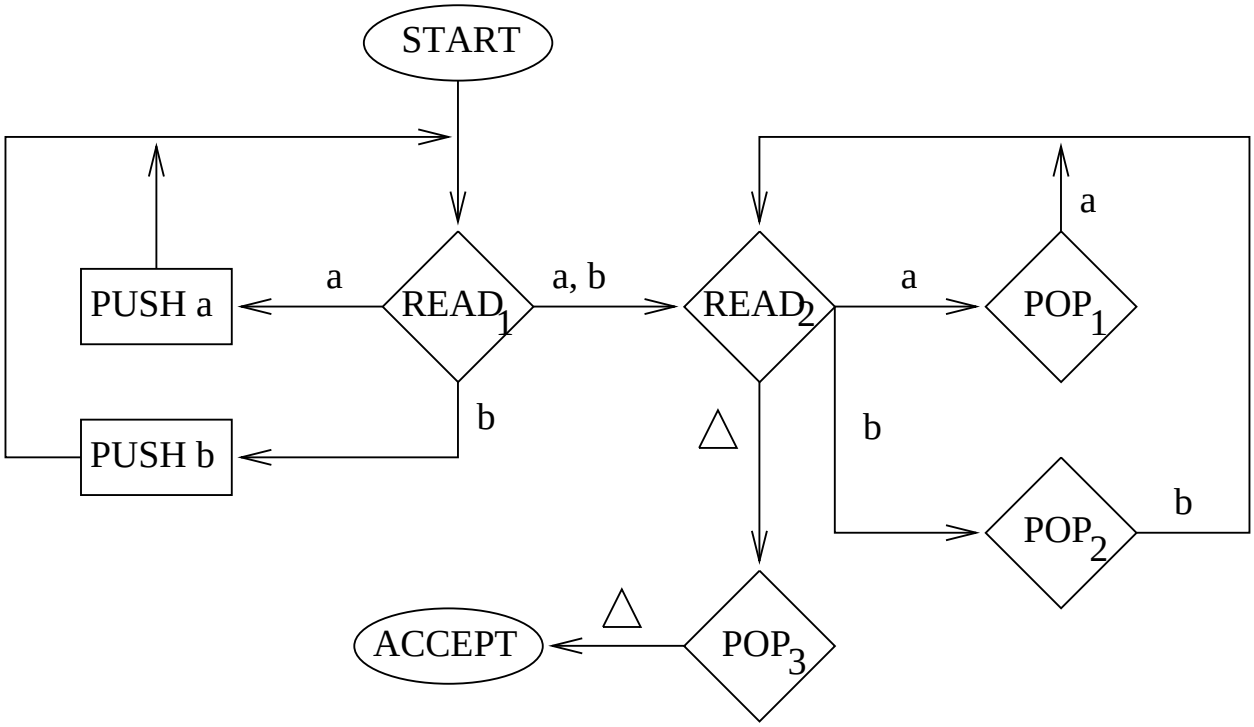


Example: Language ODDPALINDROME, which consists of all words over $\Sigma = \{a, b\}$ having odd length that are the same forwards and backwards.

$$\text{ODDPALINDROME} = \{a, b, aaa, aba, bab, bbb, aaaaa, \ldots\}$$

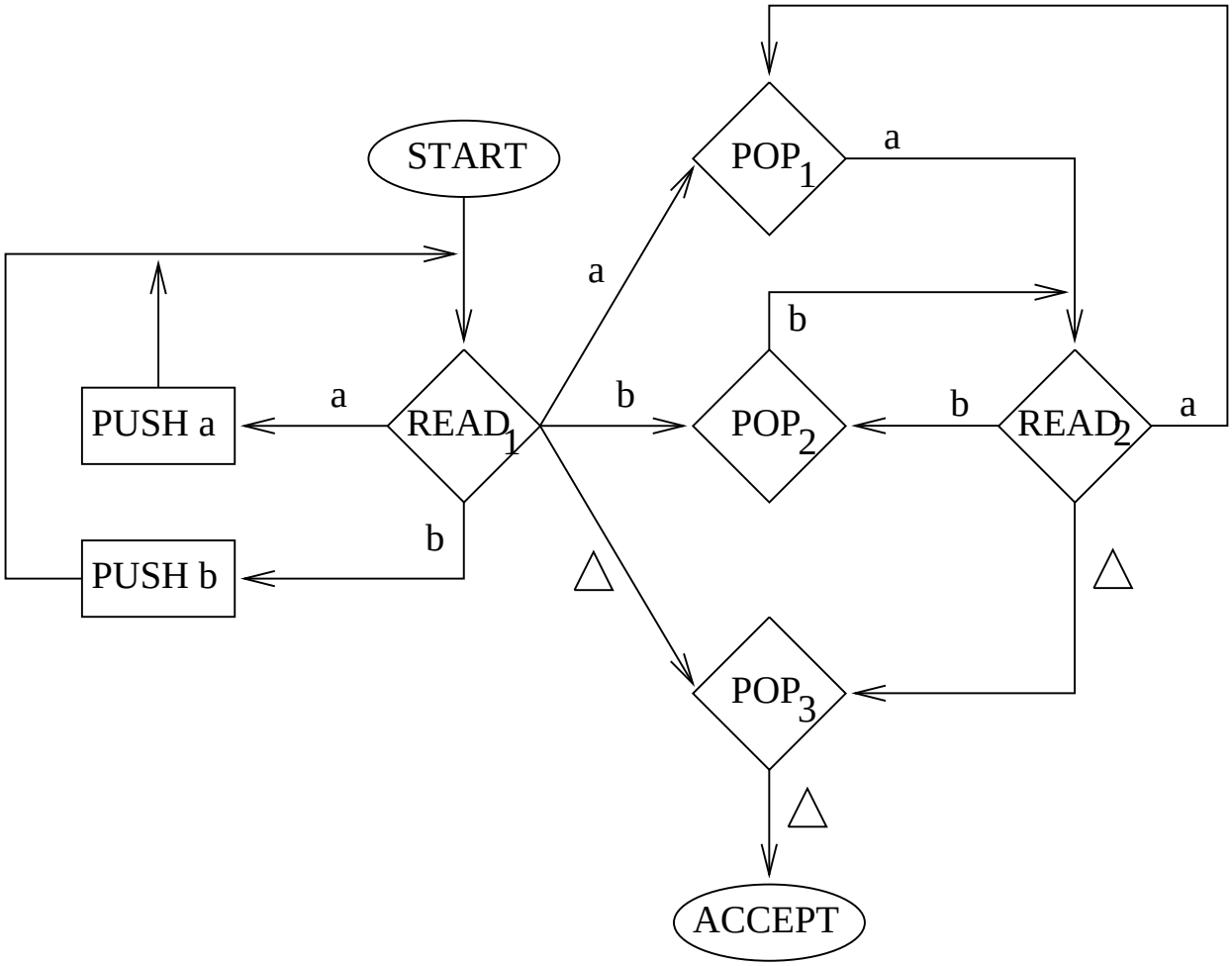
Remarks:

- For PALINDROMEX, easy to detect when at middle of word when reading TAPE since marked by X.
- For ODDPALINDROME, impossible to detect when at middle of word when reading TAPE.
- Need to use nondeterminism.



Example: Language EVENPALINDROME, which consists of all words over $\Sigma = \{a, b\}$ having even length that are the same forwards as backwords.

$$\begin{aligned} \text{EVENPALINDROME} &= \{s\,\text{reverse}(s) : s \text{ can be generated by } (\mathbf{a} + \mathbf{b})^*\} \\ &= \{\Lambda, aa, bb, aaaa, abba, baab, bbbb, aaaaaa, \ldots\} \end{aligned}$$



Suppose we had the INPUT TAPE

b	a	a	b	Δ	Δ	$\cdots$

Stack is initially empty:

Δ
Δ
$\vdots$

See what happens when we process it:

STATE	STACK	TAPE
START	$\Delta\cdots$	$baab\Delta\cdots$
READ$_1$	$\Delta\cdots$	$\not{b}aab\Delta\cdots$
PUSH b	$b\Delta\cdots$	$\not{b}aab\Delta\cdots$
READ$_1$	$b\Delta\cdots$	$\not{b}\not{a}ab\Delta\cdots$
PUSH a	$ab\Delta\cdots$	$\not{b}\not{a}ab\Delta\cdots$
READ$_1$	$ab\Delta\cdots$	$\not{b}\not{a}\not{a}b\Delta\cdots$
POP$_1$	$b\Delta\cdots$	$\not{b}\not{a}\not{a}b\Delta\cdots$
READ$_2$	$b\Delta\cdots$	$\not{b}\not{a}\not{a}\not{b}\Delta\cdots$
POP$_2$	$\Delta\cdots$	$\not{b}\not{a}\not{a}\not{b}\Delta\cdots$
READ$_2$	$\Delta\cdots$	$\not{b}\not{a}\not{a}\not{b}\not{\Delta}\Delta\cdots$
POP$_3$	$\Delta\cdots$	$\not{b}\not{a}\not{a}\not{b}\not{\Delta}\Delta\cdots$
ACCEPT	$\Delta\cdots$	$\not{b}\not{a}\not{a}\not{b}\not{\Delta}\Delta\cdots$

Alternatively, we could have processed it as follows:

STATE	STACK	TAPE
START	$\Delta\cdots$	$baab\Delta\cdots$
READ$_1$	$\Delta\cdots$	$\not{b}aab\Delta\cdots$
PUSH b	$b\Delta\cdots$	$\not{b}aab\Delta\cdots$
READ$_1$	$b\Delta\cdots$	$\not{b}\not{a}ab\Delta\cdots$
POP$_1$	$\not{b}\Delta\cdots$	$\not{b}\not{a}ab\Delta\cdots$
CRASH	$\Delta\cdots$	$\not{b}\not{a}ab\Delta\cdots$

This time the PDA crashes.

But since there is at least one way of processing the string $baaaab$ which leads to an ACCEPT state, the string is accepted by the PDA.

14.5 Formal Definition of PDA and More Examples

Definition: A *pushdown automaton* (PDA) is a collection of eight things:

1. An alphabet Σ of input letters.
2. An input TAPE (infinite in one direction), which initially contains the input string to be processed followed by an infinite number of blanks Δ
3. An alphabet Γ of STACK characters.
4. A pushdown STACK (infinite in one direction), which initially contains all blanks Δ.
5. One START state that has only out-edges, no in-edges. Can have more than one arc leaving the START state. There are no labels on arcs leaving the START state.
6. Halt states of two kinds:
   (a) zero or more ACCEPT states
   (b) zero or more REJECT states

   Each of which have in-edges but no out-edges.
7. Finitely many nonbranching PUSH states that introduce characters from Γ onto the top of the STACK.
8. Finitely many branching states of two kinds:
   (a) READ states, which read the next unused letter from TAPE and may have out-edges labeled with letters from Σ or a blank Δ. (There is no restriction on duplication of labels and no requirement that there be a label for each letter of Σ, or Δ.)
   (b) POP states, which read the top character of STACK and may have out-edges labeled with letters of Γ and the blank character Δ, with no restrictions.

Remarks:

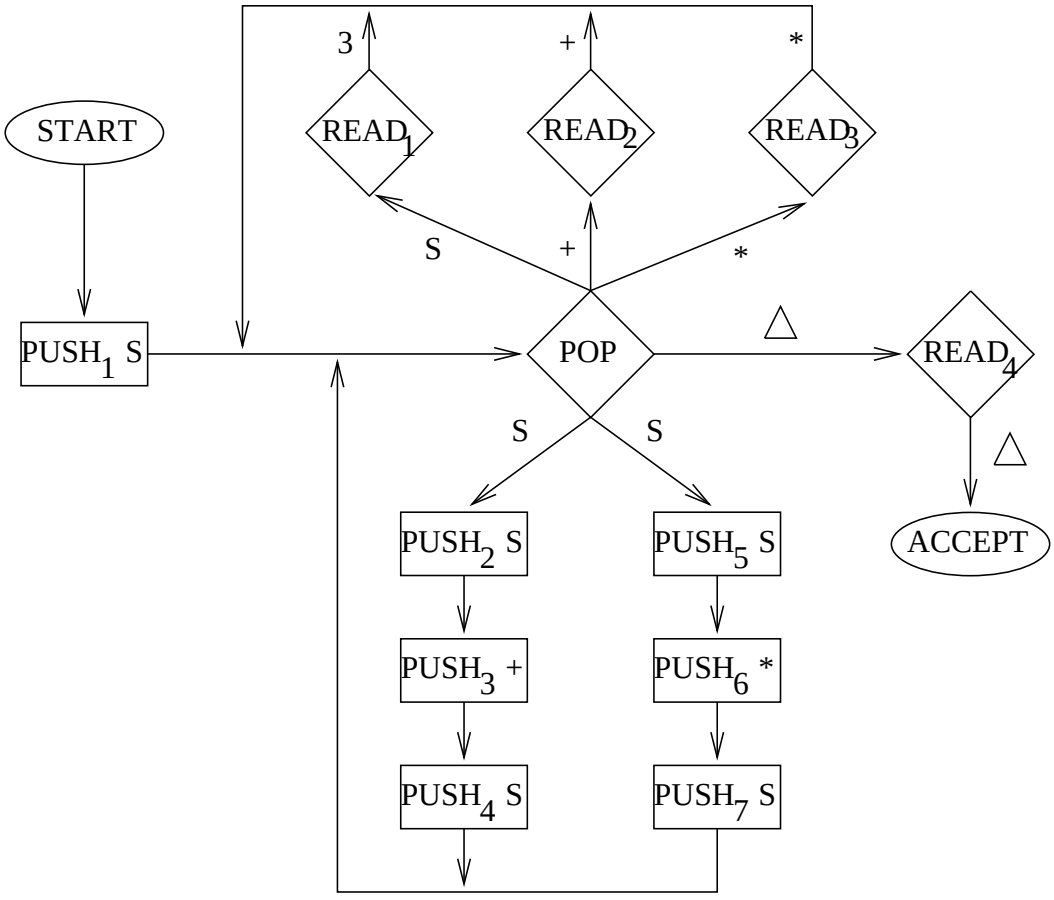
- The definition for PDA allows for nondeterminism.
- If we want to consider a PDA that does not have nondeterminism, then we will call it a deterministic PDA.

Example: CFG:

$$S \to S + S \mid S * S \mid 3$$

- terminals: $+$, $*$, 3
- nonterminals: S

(Nondeterministic) PDA:



Process $3 * 3 + 3$ on PDA, where we now erase input TAPE as we read in letters:

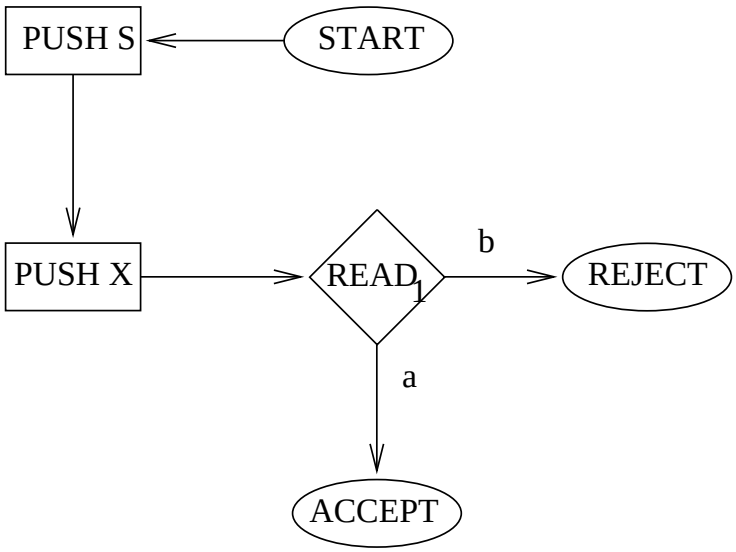
STATE	STACK	TAPE
START	$\Delta\cdots$	$3*3+3\Delta\cdots$
$\text{PUSH}_1 S$	$S\Delta\cdots$	$3*3+3\Delta\cdots$
POP	$\Delta\cdots$	$3*3+3\Delta\cdots$
PUSH_5	$S\Delta\cdots$	$3*3+3\Delta\cdots$
PUSH_6	$*S\Delta\cdots$	$3*3+3\Delta\cdots$
PUSH_7	$S*S\Delta\cdots$	$3*3+3\Delta\cdots$
POP	$*S\Delta\cdots$	$3*3+3\Delta\cdots$
READ_1	$*S\Delta\cdots$	$*3+3\Delta\cdots$
POP	$S\Delta\cdots$	$*3+3\Delta\cdots$
READ_3	$S\Delta\cdots$	$3+3\Delta\cdots$
POP	$\Delta\cdots$	$3+3\Delta\cdots$
PUSH_2	$S\Delta\cdots$	$3+3\Delta\cdots$
PUSH_3	$+S\Delta\cdots$	$3+3\Delta\cdots$
PUSH_4	$S+S\Delta\cdots$	$3+3\Delta\cdots$
POP	$+S\Delta\cdots$	$3+3\Delta\cdots$
READ_1	$+S\Delta\cdots$	$+3\Delta\cdots$
POP	$S\Delta\cdots$	$+3\Delta\cdots$
READ_2	$S\Delta\cdots$	$3\Delta\cdots$
POP	$\Delta\cdots$	$3\Delta\cdots$
READ_1	$\Delta\cdots$	$\Delta\cdots$
POP	$\Delta\cdots$	$\Delta\cdots$
READ_4	$\Delta\cdots$	$\Delta\cdots$
ACCEPT	$\Delta\cdots$	$\Delta\cdots$

14.6 Some Properties of PDA

Theorem 28 *For every regular language L, there is some PDA that accepts it.*

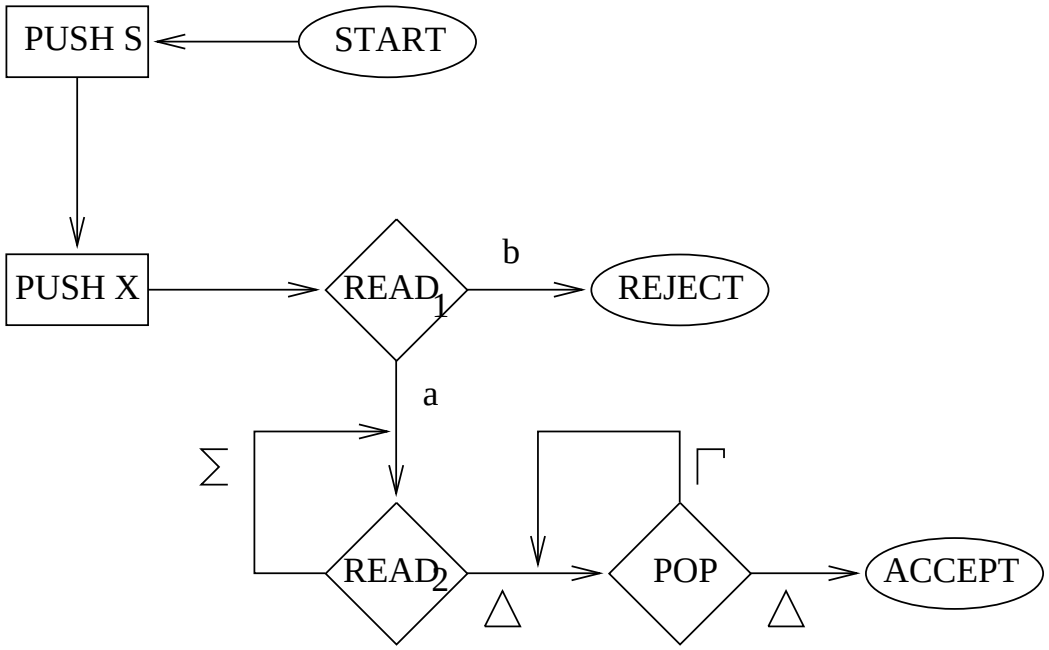
Note that PDA can reach ACCEPT state and still have non-blank letters on TAPE and/or STACK.

Example:



Theorem 29 *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.*

Proof. Can convert above PDA into equivalent one below:


PUSH S
START
PUSH X
READ1
b
REJECT
a
Σ
READ2
POP
ACCEPT
Γ
Δ
Δ

■