Chapter 14: Pushdown Automata *

Peter Cappello
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106
cappello@cs.ucsb.edu

• The corresponding textbook chapter should be read before attending this lecture.

• These notes are not intended to be complete. They are supplemented with figures, and other material that arises during the lecture period in response to questions.

*Based on Theory of Computing, 2nd Ed., D. Cohen, John Wiley & Sons, Inc.
A New Format for Finite Automata

We give an alternate format for describing FA:

- The finite sequence of symbols to be read, the input word, resides on an input tape, which extends infinitely to the right, and which is divided into infinitely many cells.
  - The 1st symbol is in the leftmost cell.
  - The 2nd symbol is in the cell to the right, and so on.
  - All cells beyond those that have an input symbol contain a blank, denoted by $\Delta$.
  - The machine starts in a position ready to read the leftmost symbol on the input tape.
– After it reads this symbol, it moves in position (1 cell to the right) to read the next symbol.

Give a picture of an input tape with the word “abba”.

• We introduce the following 3 new diagrammatic symbols: Draw the START, ACCEPT, and REJECT symbols.
  – A START is like the - state, connected to a state by a $\Lambda$-edge.
  – An ACCEPT is a dead-end (once entered, never exited) final state.
  – A REJECT is a dead-end non-final state.

Accept and Reject states are called **halt** states.

• We introduce the READ symbol. Draw a picture of this.
  It denotes reading 1 symbol on the input tape (and positioning to read the next symbol).
• Depict 2 examples:
  – an FA that accepts $(a + b)^*a$.
  – an FA that accepts $(a + b)^*aa(a + b)^*$. 
Adding a Stack

• Adding a stack to our finite automaton results machine is called a pushdown automaton (PDA).

• It has:
  – a PUSH s operation Draw a picture of this.
  – a POP operation. Draw a picture of this.
    Popping the empty stack yields a Δ.

• Example: What language does the following PDA accept?

• The alphabet of stack symbols need not be the input alphabet.

• A PDA with only 1 arc for each symbol (input and stack) is a deterministic PDA (DPDA).
• If a PDA has either a READ or POP state with $> 1$ arc for some symbol, it is a **non-deterministic PDA (NPDA)**.

• We may omit arcs leading out of a READ or POP state. When such an arc is omitted, it means that the machine crashes (i.e., goes to a REJECT state).

• Example: Here is a DPDA for:

$$X - centered\ palindromes = \{wXw^R \mid w \in (a + b)^*\},$$

where $w^R$ means the reverse of $w$, and the input alphabet $= \{a, b, X\}$.
• Example: Here is an NPDA for:

Odd length palindromes = \{ w(a + b)w^R \mid w \in (a + b)^* \}, where 
\( w^R \) means the reverse of \( w \).

The language above is contained in the language of palindromes.

• Example: Here is an NPDA for:

Even length palindromes = \{ ww^R \mid w \in (a + b)^* \}, where \( w^R \) means the reverse of \( w \).

The language above is contained in the language of palindromes.
Defining the PDA

Definition

A pushdown automaton, PDA, is a collection of 8 things:

1. An alphabet $\Sigma$ of input symbols.
2. An input TAPE (infinite in 1 direction). The input word starts in the leftmost cell. The rest of the TAPE is blank.
3. An alphabet $\Gamma$ of stack symbols.
4. A stack (infinite in 1 direction), initially blank.
5. One START state that has only out-edges.
6. Halt states (ACCEPT & REJECT). They only have in-edges.
7. Finitely many non-branching PUSH states.

8. Finitely many branching states of 2 kinds:

   • States that READ the next input symbol, with out-edges labelled with the blank character, $\Delta$, and elements of $\Sigma$. There need not be an arc for each such symbol.

   • States that POP the stack, with out-edges labelled with the blank character, $\Delta$, and elements of $\Gamma$. There need not be an arc for each such symbol.

• All the states above form a connected directed graph.

• To **run** the PDA on an input:

  – Beginning from the START state, follow unlabelled edges and labelled edges that apply to produce a path through the graph.

  – This path ends in either:
* a halt state
* a crash when there is no edge corresponding to the symbol read/popped.

- An input with some path to an ACCEPT is **accepted**.
- The set of strings accepted by a PDA is called the **language accepted** by the PDA, or the **language recognized** by the PDA.
Example

Consider the language generated by the CFG

\[ S \rightarrow S + S \mid S * S \mid 4. \]

The following PDA accepts this language. Draw on board.
Theorem

For every regular language $L$, there is a PDA that accepts it.

Proof

The notation that we introduced without the stack, is an alternate notation for finite automata. Thus, it is capable of expressing any finite automaton.
Theorem

Given a PDA $A$, there is a PDA $B$ that accepts the same language. Moreover, when an ACCEPT state in $B$ is entered, all the input has been read and stack is empty.

Proof

Replace each of $A$’s ACCEPT states with the following routine:
The arcs into the $A$’s ACCEPT state go into a new READ state in $B$. This READ state’s out-edges are self-loops, except on $\Delta$.
The $\Delta$ arc is to a POP state.
The POP state out-edges are self-loops, except on $\Delta$.
The $\Delta$ arc is to an ACCEPT state.
If we replace $A$’s ACCEPT states with this sequence, when the new ACCEPT state is reached, all the input has been read and the stack is empty.