Pushdown Automata

- A pushdown automata (PDA) is essentially:
  - An NFA with a stack
  - A "move" of a PDA will depend upon
    - Current state of the machine
    - Current symbol being read in
    - Current symbol popped off the top of the stack
  - With each "move", the machine can
    - Move into a new state
    - Push symbols on to the stack

Pushdown Automata

The stack

- The stack has its own alphabet
- Included in this alphabet is a special symbol used to indicate an empty stack. (z)
- Note that the basic PDA is non-deterministic!

Pushdown Automata

Let’s formalize this:

A pushdown automata (PDA) is a 7-tuple:

\[ M = (Q, \Sigma, \Gamma, \delta, q_0, z, F) \]

- \( Q \): finite set of states
- \( \Sigma \): tape alphabet
- \( \Gamma \): stack alphabet (may have symbols in common w/ \( \Sigma \))
- \( q_0 \in Q \): start state
- \( z \in \Gamma \): initial stack symbol
- \( F \subseteq Q \): set of accepting states
- \( \delta \): transition function

Pushdown Automata

About this transition function \( \delta \):

- During a move of a PDA:
  - At most one character is read from the input tape
  - \( \lambda \) transitions are okay
  - The topmost character is popped from the stack
  - The machine will move to a new state based on:
    - The character read from the tape
    - The character popped off the stack
    - The current state of the machine
  - 0 or more symbols from the stack alphabet are pushed onto the stack.
Pushdown Automata

Formally:
\[ \delta: Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow (\text{finite subsets of } Q \times \Gamma^*) \]

- Domain:
  - \( Q = \) state
  - \( \Sigma \cup \{\lambda\} = \) symbol read off tape
  - \( \Gamma = \) symbol popped off stack
- Range:
  - \( Q = \) new state
  - \( \Gamma^* = \) symbols pushed onto the stack

Example:
\[ \delta(q, a, a) = (p, aa) \]
Meaning:
- When in state \( q \),
- Reading in an \( a \) from the tape
- With an \( a \) popped off the stack
- The machine will
  - Go into state \( p \)
  - Push the string "aa" onto the stack

Configuration of a PDA

Gives the current "configuration" of the machine

\( (p, x, \alpha) \) where
- \( p = \) the current state
- \( x = \) a string indicating what remains to be read on the tape
- \( \alpha = \) the current contents of the stack.

Move of a PDA:

We can describe a single move of a PDA:
\[ (q, x, \alpha) \rightarrow (p, y, \beta) \]
If:
- \( x = ay, \alpha = \gamma, \beta = YX \)
  - And
- \( \delta(q, x, \gamma) \) includes \((p, Y)\)
- \( \delta(q, \lambda, \gamma) \) includes \((p, Y)\) and \( x = y \).

Strings accepted by a PDA by Final State

- Start at \((q_0, x, \lambda)\)
- X on the input tape
- Empty stack
- End with \((q, \lambda, \beta)\)
  - End in an accepting state \((q \in F)\)
  - All characters of \( x \) have been read
  - Some string on the stack (doesn't matter what).
Strings accepted by a PDA (Final State)

Let \( M = (Q, \Sigma, \Gamma, \delta, q_0, z, F) \) be a PDA

\( x \) is accepted by \( M \) if

\[ (q_0, x, z) \xrightarrow{a^*} (q, \lambda, \beta) \]

Where

- \( q \in F \)
- \( \beta \in \Gamma^* \)

The language accepted by a PDA

Let \( M = (Q, \Sigma, \Gamma, q_0, z, F, \delta) \) be a PDA

The language accepted by \( M \) by final state,

Denoted \( L(M) \) is

The set of all strings \( x \) that are accepted by \( M \) by final state

Let’s look at an example:

\( L = \{ xc x r | x \in \{ a,b \}^* \} \)

Basic idea for building a PDA

Read chars off the tape until you reach the ‘c’.

As you read chars push them on the stack

After reading the c, match the chars read with the chars popped off the stack until all chars are read

If at any point the char read does not match the char popped, the machine “crashes”

Let’s look at an example:

\( L = \{ xc x r | x \in \{ a,b \}^* \} \)

The PDA will have 4 states

- State 0 (initial) : reading before the ‘c’
- State 1: read the ‘c’
- State 2 :read after ‘c’, comparing chars
- State 3: (accepting): move only after all chars read and stack empty

Let’s look at an example:

\( L = \{ xc x r | x \in \{ a,b \}^* \} \)

Transition for abcba

\( (q_0, abcba, Z) \rightarrow (q_0, bcba, a) \) // push a

\( (q_0, cba, ba) \) // push b

\( (q_1, ba, ba) \) // goto 1

\( (q_2, ba, ba) \) // \( \lambda \) trans

\( (q_2, a, a) \) // pop b

\( (q_2, \lambda, Z) \) // pop a

\( (q_3, \lambda, Z) \) // Accept!
### Pushdown Automata

- I bet you’re wondering if JFLAP can handle PDAs!
  - Yes, it can...
  - Let’s take a look.

---

### Pushdown Automata

- Let’s look at another example:
  - \( L = \{ xx^r \mid x \in \{a,b\}^* \} \)

  - Basic idea for building a PDA
   - Much like last example, except
     - This time we don’t know when to start popping and comparing
     - Since PDAs are non-deterministic, this is not a problem

---

### Pushdown Automata

- Let’s look at an example:
  - \( L = \{ xx^r \mid x \in \{a,b\}^* \} \)

  ![Diagram](image)

---

### PDA Example

- Let’s see a bad transition set for abba
  - \((q_0, abba, z) \rightarrow (q_0, bba, a)\) // push a
  - \(\rightarrow (q_1, ba, ba)\) // push b
  - \(\rightarrow (q_2, a, bba)\) // push b
  - \(\rightarrow (q_2, a, bba)\) // \(\varepsilon\) trans
  - Nowhere to go // Reject!
**PDA Example**

- Let’s see a good transition set for abba
  - \((q_0, \text{abba}, z) \rightarrow (q_0, \text{bbba}, a)\) // push a
  - \((q_0, \text{ba}, \text{ba})\) // push b
  - \((q_0, \text{ba}, \text{ba})\) // \(\epsilon\) trans
  - \((q_0, a, a)\) // pop b
  - \((q_1, \text{ba}, \lambda, Z)\) // pop a
  - \((q_1, \lambda, Z)\) // Accept!

**Pushdown Automata**

- “Let’s go to the video tape”
- Actually JFLAP...

---

**Pushdown Automata**

- Strings accepted by a PDA by Final State
  - Start at \((q_0, x, z)\)
    - Start state \(q_0\)
    - \(x\) on the input tape
    - Empty stack
  - End with \((q, \lambda, \beta)\)
    - End in an accepting state \((q \in \mathcal{F})\)
    - All characters of \(x\) have been read
    - Some string on the stack (doesn’t matter what).

**Pushdown Automata**

- Strings accepted by a PDA by Empty Stack
  - Start at \((q_0, x, z)\)
    - Start state \(q_0\)
    - \(x\) on the input tape
    - Empty stack
  - End with \((q, \lambda, \lambda)\)
    - End in any state
    - All characters of \(x\) have been read
    - Stack is empty

---

**Pushdown Automata**

- Strings accepted by a PDA (Final State)
  - Let \(M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)\) be a PDA
  - \(x\) is accepted by \(M\) if
    - \((q_0, x, z) \rightarrow (q, \lambda, \beta)\)
    - Where
      - \(q \in Q\)
      - \(\beta \in \Gamma^*\)

---

**Pushdown Automata**

- Strings accepted by a PDA (Empty Stack)
  - Let \(M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)\) be a PDA
  - \(x\) is accepted by \(M\) if
    - \((q_0, x, z) \rightarrow (q, \lambda, \lambda)\)
    - Where
      - \(q \in Q\)
Pushdown Automata

- The language accepted by a PDA
  - Let M = (Q, Σ, Γ, q₀, z, F, δ) be a PDA
  - The language accepted by M by final state,
    - Denoted L(M) is
    - The set of all strings x that are accepted by M by final state
  - The language accepted by M by empty stack,
    - Denoted N(M) is
    - The set of all strings x that are accepted by M by empty stack
  - We will show that all languages accepted by a PDA by final state will be accepted by an equivalent PDA by empty stack and visa versa

Final State vs. Empty Stack

- The two means by which a PDA can accept are equivalent wrt the class of languages accepted
  - Given a PDA M such that L = L(M), there exists a PDA M’ such that L = N(M’)
  - Given a PDA M such that L = N(M), there exists a PDA M’ such that L = L(M’)

Final State → Empty Stack

- Final State → Empty Stack
  - Given a PDA Pₖ = (Q, Σ, Γ, δₖ, q₀, z, F) and L = L(Pₖ) then there exists a PDA Pₙ such that L = N(Pₙ)
  - We will build such a PDA

Accept by Empty Stack

- Basic idea
  - Transitions of Pₙ will mimic those of Pₖ
  - Create a new state in Pₙ that will empty the stack.
  - The machine can move into this new state whenever the machine is in an accepting state of Pₖ

- Final State → Empty Stack
  - We must be careful though
    - Pₖ may crash when the stack is empty
    - In those cases we need to assure that Pₙ does not accept
    - To solve this:
      - Create a new empty stack symbol X₀ which is placed on the stack before Pₖ’s empty stack marker (z)
      - z will only be popped by the new “stack emptying state
      - The first move of Pₙ will be to place zX₀ on Pₙ stack.
Accept by Empty Stack

- Final State $\rightarrow$ Empty Stack

Empty Stack $\rightarrow$ Final State

- Given a PDA $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0)$ and $L = N(P_N)$ then there exists a PDA $P_F$ such that $L = L(P_F)$

- We will build such a PDA
- Actually, you will...Exercise 17

Reality Check

- Pushdown Automata
  - NFAs with a stack
  - Move depends on tape symbol, state, and top of stack.
  - Move involves popping stack, moving to new state and pushing onto stack.
  - Basic PDA is non-deterministic.

- Accept by final state
- Accept by empty stack

Next time

- PDAs...the perfect machine for CFLs...

- Questions?