Pushdown Automata

Homework

• Homework #3
  – Will return Tuesday
• Homework #4
  – Early submission: Due today
    • Will return on Tuesday
  – Regular submission: Due Tuesday
• Homework #5
  – PDAs
  – Will be assigned Tues, October 19.

Final Exam

• Good news – bad news
  – Good news
    • Final exam schedule is out
  – Even better news
    • CS Theory Exam is on Thursday, Nov 18th
  – Bad news
    • Exam is at 8am!!!!
• Thursday, November 18th, 8am, 70-3435

Plan

• Today
  – Pushdown Automata
  – CFG (Chapter 5) Problem Session
• Tuesday
  – CFG <-> PDA Equivalence
  – Midterm Review
• Thursday
  – Midterm

Languages

• Recall.
  – What is a language?
  – What is a class of languages?

Context Free Languages

• Context Free Languages(CFL) is the next class of languages outside of Regular Languages:
  – Means for defining: Context Free Grammar
  – Machine for accepting: Pushdown Automata
Plan for today

- Introduction to Pushdown Automata

Pushdown Automata

- A pushdown automata (PDA) is essentially:
  - An NFA-ε 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. (Z₀)
    - This special symbol should not be removed from the stack.
  - Note that the basic PDA is non-deterministic!

Pushdown Automata

- Let’s formalize this:
  - A pushdown automata (PDA) is a 7-tuple:
    - $M = (Q, \Sigma, \Gamma, \delta, q₀, Z₀, F)$ where
      - $Q$ = finite set of states
      - $\Sigma$ = tape alphabet
      - $\Gamma$ = stack alphabet (may have symbols in common w/ $\Sigma$)
      - $q₀ \in Q$ = start state
      - $Z₀ \in \Gamma$ = 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
      - ε 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:
  - $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow (\text{finite subsets of } Q \times \Gamma^*)$
  - Domain:
    - $Q =$ state
    - $(\Sigma \cup \{\varepsilon\}) =$ symbol read off tape
    - $\Gamma =$ symbol popped off stack
  - Range:
    - $Q =$ new state
    - $\Gamma^* =$ symbols pushed onto the stack

Pushdown Automata

- 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

Pushdown Automata

- Configuration of a PDA
  - Gives the current “configuration” of the machine
  - $(p, x, \alpha)$ where
    - $p =$ current state
    - $x =$ string indicating what remains to be read on the tape
    - $\alpha =$ current contents of the stack

Pushdown Automata

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

Pushdown Automata

- Strings accepted by a PDA by Final State
  - Start at $(q_0, x, Z_0)$
    - Start state $q_0$
    - $X$ on the input tape
    - Empty stack
  - End with $(q, \varepsilon, \beta)$
    - End in an accepting state $(q \in F)$
    - All characters of $x$ have been read
    - Some string on the stack (doesn’t matter what).
Pushdown Automata

• Strings accepted by a PDA (Final State)
  – Let \( M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F) \) be a PDA
  – \( x \) is accepted by \( M \) if
    • \( (q_0, x, Z_0) \xrightarrow{a^*} (q, \varepsilon, \beta) \)
    • Where
      – \( q \in A \)
      – \( \beta \in \Gamma^* \)

Pushdown Automata

• Strings accepted by a PDA (Empty Stack)
  – Let \( M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F) \) be a PDA
  – \( x \) is accepted by \( M \) if
    • \( (q_0, x, Z_0) \xrightarrow{a^*} (q, \varepsilon, \varepsilon) \)
    • Where
      – \( q \in Q \)

Pushdown Automata

• The language accepted by a PDA
  – Let \( M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, A) \) 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 next week that all languages accepted by a PDA by final state will be accepted by an equivalent PDA by empty stack and visa versa

Pushdown Automata

• Let’s look at an example:
  – \( L = \{ x \varepsilon x \varepsilon^* \mid x \in \{a, b \}^* \} \)
    • 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”

Pushdown Automata

• Let’s look at an example:
  – \( L = \{ x \varepsilon x \varepsilon^* \mid x \in \{a, b \}^* \} \)
    • 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
Pushdown Automata

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

- Transition for \( abcba \)
  - \((q_0, abcba, Z) \mapsto (q_0, cbba, a) \) // push a
  - \(\mapsto (q_0, cb, ba) \) // push b
  - \(\mapsto (q_1, b, ba) \) // goto 1
  - \(\mapsto (q_2, b, ba) \) // \(\epsilon\) trans
  - \(\mapsto (q_2, a, a) \) // pop b
  - \(\mapsto (q_3, a, Z) \) // pop a
  - \(\mapsto (q_3, \epsilon, Z) \) // Accept!

PDA Example

- Transition for \( abcb \)
  - \((q_0, abcb, Z) \mapsto (q_0, bcb, a) \) // push a
  - \(\mapsto (q_0, cb, ba) \) // push b
  - \(\mapsto (q_1, b, ba) \) // goto 1
  - \(\mapsto (q_2, b, ba) \) // \(\epsilon\) trans
  - \(\mapsto (q_2, a, a) \) // pop b
  - Nowhere to go // Reject!

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:
  - \( L = \{ x x^r \mid x \in \{ a,b \}^* \} \)

  - 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:
  - $L = \{ xx^r | x \in \{a, b\}^* \}$

\[
\begin{array}{l}
q_0 \quad q_1 \\
\text{b, Z} / bZ_a \\
\text{s, Z} / aZ_a \\
\text{b, b / b} \\
\end{array}
\]

PDA Example

- Let’s see a bad transition set for abba
  - $(q_0, abba, Z) \mapsto (q_0, bba, a)$ // push a
  - $\mapsto (q_0, ba, ba)$ // push b
  - $\mapsto (q_1, a, bba)$ // push b
  - $\mapsto (q_1, a, bba)$ // $\varepsilon$ trans
  - Nowhere to go // Reject!

PDA Example

- Let’s see a good transition set for abba
  - $(q_0, abba, Z) \mapsto (q_0, bba, a)$ // push a
  - $\mapsto (q_0, ba, ba)$ // push b
  - $\mapsto (q_1, ba, ba)$ // $\varepsilon$ trans
  - $\mapsto (q_1, a, a)$ // pop b
  - $\mapsto (q_1, a, Z)$ // pop a
  - $\mapsto (q_2, Z)$ // Accept!

Deterministic PDAs

- As mentioned before
  - Our basic PDA in non-deterministic
  - We can define a Deterministic PDA (DPDA) as follows:
    \[\text{Let } M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F) \text{ be a PDA}\]
    - $M$ is deterministic if:
      1. $\delta(q, a, X)$ has at most one element
      2. If $\delta(q, c, X) \neq \emptyset$ then $\delta(q, a, X) = \emptyset$ for all $a \in \Sigma$

Deterministic PDAs

- In other words:
  - There is no configuration where the machine has a “choice” of moves
    - Each transition has at most 1 element.
    - If you can make a $\varepsilon$ -transition from a state with a given symbol on the stack, you cannot make that same transition on any tape input symbol.
Deterministic PDAs

- A language $L$ is a **deterministic context-free language (DCFL)** if there is a DPA that accepts $L$.

PDA Example

- Example:
  - $L = \{ x \in \{a, b\}^* \mid n_a(x) > n_b(x) \}$
  
  - First using a PDA:
    - Let the stack store the “excess” of one symbol over another.
      - If more $a$'s have been read than $b$'s, $a$'s will be on the stack, and vice versa.
      - If $a$ is on the stack and you read a $b$, simple match the $a$ with the $b$.
      - If $a$ is on the stack and you read an $a$, we have one more extra $a$ – Push it on the stack.
    - An empty stack means the number of $a$’s and $b$’s are equal.

- The PDA will have 2 states:
  - State 0 (start) : where all the work gets done
  - State 1 (accepting) : one you’re in here, the machine stops.
  - The machine can “choose” to go into state 1 on a $\epsilon$ transition whenever an $a$ is on the stack.

- Removing the non-determinism:
  - Let the stack store $1$ minus the “excess” of one symbol over another.
  - The state will determine whether you have excess $a$’s or excess $b$’s.

Non-determinism
PDA Example

- Example:
  \[ L = \{ x \in \{ a, b \}^* | n_a(x) > n_b(x) \} \]

Now you might be wondering...

- We know that all DCFLs are CFLs.

It can be shown...

- That the language \( \text{pal} \):
  \[ \text{pal} = \{ x \in \{ a, b \}^* | x = x^r \} \]

  - Cannot be accepted by any DPDA.

It can also be shown

- That all regular languages can be accepted by a DPDA.
  - Since an DFA is essentially a DPDA that doesn’t make use of the stack.

Now our picture looks like

Why DPDAs are important

- A compiler may wish to implement a PDA in software to parse a program given by a given grammar.
- DPDAs and ambiguity
  - If \( L \) can be accepted by a DPDA, then \( L \) can be expressed by an unambiguous CFG.
  - Not visa versa.
Determinism vs. Non-Determinism

- Comparing FAs and PDAs
  - DPDAs allow for ε-transitions
  - DPDAs allow for no moves
  - FAs and NFAs are equivalent
  - PDAs and DPDAs are not equivalent

Questions

Pushdown Automata

- Questions?

- Let’s break