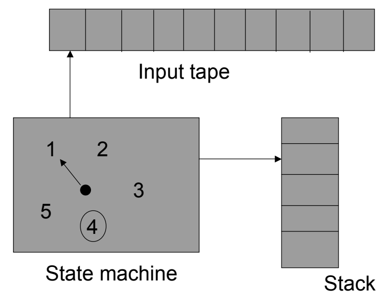
Context Free Languages IV

Pushdown Automata

Pushdown Automata



Plan for today

- Introduction to Pushdown Automata

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_0)
    - This special symbol should not be removed from the stack.
- Note that the basic PDA is non-deterministic!

Pushdown Automata

- A pushdown automata (PDA) is essentially:
  - An NDFA- Λ 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

- Let's formalize this:
  - A pushdown automata (PDA) is a 7-tuple:
    - $M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$ where
      - Q = finite set of states
      - Σ = tape alphabet
      - Γ = stack alphabet (may have symbols in common w/ Σ)
      - $q_0 \in Q$ = start state
      - $Z_0 \in \Gamma$ = initial stack symbol
      - $A \subseteq Q$ = set of accepting states
      - δ = transition function

Pushdown Automata

- About this transition function δ:
  - During a move of a PDA:
    - At most one character is read from the input tape
      - Λ transitions are okay
    - The topmost character is popped from the stack
      - Unless it is Z_0
    - 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

- Configuration of a PDA
  - Gives the current "configuration" of the machine
  - (p, x, α) where
    - p is the current state
    - x is a string indicating what remains to be read on the tape
    - α is the current contents of the stack.

Pushdown Automata

- Formally:
  - $\delta: Q \times (\Sigma \cup \{\Lambda\}) \times \Gamma \to$ (finite subsets of $Q \times \Gamma^*$)
  - Domain:
    - Q = state
    - $(\Sigma \cup \{\Lambda\})$ = symbol read off tape
    - Γ = symbol popped off stack
  - Range
    - Q = new state
    - Γ^* = symbols pushed onto the stack

Pushdown Automata

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

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

- Moves of a PDA
  - We can write:
    - $(q, x, \alpha) \mapsto^* (p, y, \beta)$
    - If
      - You can get from one configuration to the other by applying 0 or more moves.

Pushdown Automata

- Strings accepted by a PDA
  - Let $M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$ be a PDA
  - x is accepted by M if
    - $(q_0, x, Z_0) \mapsto^* (q, \Lambda, \beta)$
    - Where
      - $q \in A$
      - $\beta \in \Gamma^*$

Pushdown Automata

- Strings accepted by a PDA
  - Start at (q_0, x, Z_0)
    - Start state q_0
    - X on the input tape
    - Empty stack
  - End with (q, Λ, β)
    - End in an accepting state
    - All characters of x have been read
    - Some string on the stack (doesn't matter what).
  - Acceptance by "final state"

Pushdown Automata

- The language accepted by a PDA
  - Let $M = (Q, \Sigma, \Gamma, q_0, Z_0, A, \delta)$ be a PDA
  - The language accepted by M,
    - Denoted L(M) is
    - The set of all strings x that are accepted by M.

Pushdown Automata

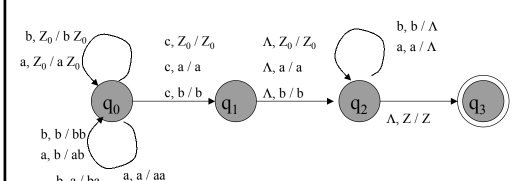
- Let's look at an example:
  - $L = \{ xcx^r \mid x \in \{ a,b \}^* \}$
  - Basic idea for building a PDA
    - Read chars off the tape until you reach the 'c'.
    - As you read chars push them on the stack
    - After reading the c, match the chars read with the chars popped off the stack until all 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:
  - $L = \{ xcx^r \mid x \in \{ a,b \}^* \}$
  - 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:
  - $L = \{ xcx^r \mid x \in \{ a,b \}^* \}$



PDA Example

- Transition for abcba
  - (q_0, abcba, Z) $\mapsto$ (q_0, bcba, a) // push a
  - $\mapsto$ (q_0, cba, ba) // push b
  - $\mapsto$ (q_1, ba, ba) // goto 1
  - $\mapsto$ (q_2, ba, ba) // Λ trans
  - $\mapsto$ (q_2, a, a) // pop b
  - $\mapsto$ (q_2, Λ, Z) // pop a
  - $\mapsto$ (q_3, Λ, 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) // Λ trans
  - $\mapsto$ (q_2, Λ, 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

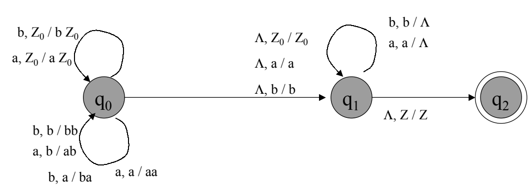
- Let's look at another example:
  - L = { xx^r | x $\in$ { a,b }* }
  - Basic idea for building a PDA
    - Much like last example, except
      - This time we don't know when to start popping and comparing
      - Since PDAs are non-deterministic, this is not a problem

Pushdown Automata

- Let's look at another example:
  - L = { xx^r | x $\in$ { a,b }* }
  - The PDA will have 3 states
    - State 0 (initial) : reading before the center of string
    - State 1: read after center of string, comparing chars
    - State 2 (accepting): after all chars read, stack should be empty
  - The machine can choose to go from state 0 to state 1 at any time:
    - Will result in many "wrong" set of moves
    - All you need is one "right" set of moves for a string to be accepted.

Pushdown Automata

- Let's look at an example:
  - L = { xx^r | x $\in$ { a,b }* }



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
  - $\mapsto (q_0, a, bba)$ // push b
  - $\mapsto (q_1, a, bba)$ // Λ trans
  - Nowhere to go // Reject!

Pushdown Automata

- Questions?

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
  - $\mapsto (q_1, ba, ba)$ // Λ trans
  - $\mapsto (q_1, a, a)$ // pop b
  - $\mapsto (q_1, \Lambda, Z)$ // pop a
  - $\mapsto (q_2, \Lambda, Z)$ // Accept!

Deterministic PDAs

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

Pushdown Automata

- "Let's go to the video tape"
  - Actually JFLAP…

Deterministic PDAs

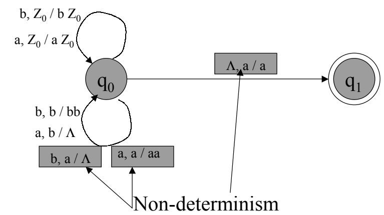
- In other words:
  - There is no configuration where the machine has a "choice" of moves
    - Each transition has at most 1 element.
    - If you can make a Λ-transition from a state with a given symbol on the stack,
      - You cannot make that same transition on any tape input symbol.

Deterministic PDAs

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

PDA Example

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



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 via 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.

PDA Example

- Let's try on JFLAP

PDA Example

- Example:
  - $L = \{ x \in \{ a, b \}^* \mid n_a(x) > n_b(x) \}$
  - 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 Λ transition whenever an a is on the stack.

PDA Example

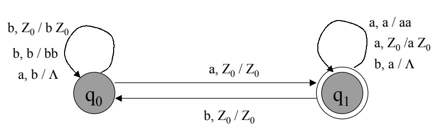
Example:
- $L = \{ x \in \{ a, b \}^* \mid n_a(x) > n_b(x) \}$
- Removing the non-determinism :
  - Let the stack store 1 minus the "excess" of one symbol over another
  - The state will determine whether you have excess a's or excess b's

PDA Example

- Example:
  - $L = \{ x \in \{ a, b \}^* \mid n_a(x) > n_b(x) \}$
  - The PDA will have 2 states:
    - State 0 (start) : when $n_a(x) \le n_b(x)$
      - Equality or surplus of b's
    - State 1 (accepting) : when $n_a(x) > n_b(x)$
      - Surplus of a's

PDA Example

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

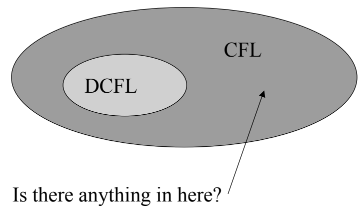


PDA Example

- Let's try on JFLAP

Now you might be wondering…

We know that all DCFLs are CFLs



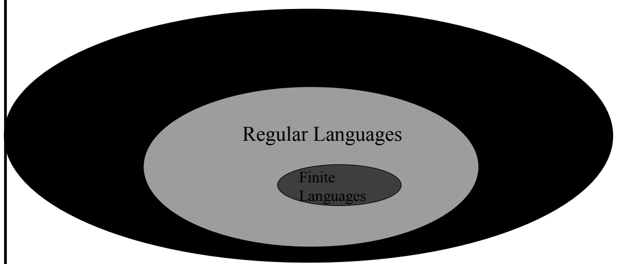
It can be shown…

- That the language pal:
  - $pal = \{ x \in \{ a, b \}^* \mid x = x^r \}$
- Cannot be accepted by any DPDA.
- See Theorem 7.1 in book for proof.

It can also be shown

- That all regular languages can be accepted by a DPDA.
  - Since an FA (deterministic) is essentially a DPDA that doesn't make use of the stack.

Now our picture looks like



Next time

- Equivalence of CFLs and PDAs

Determinism vs. Non-Determinism

- Comparing FAs and PDAs
  - DPDAs allow for Λ-transitions
  - DPDAs allow for no moves
  - FAs and NDFAs are equivalent
  - PDAs and DPDAs are not equivalent

Summary

- Pushdown Automata
  - NDFA-Λs with a stack
  - Deterministic PDAs
    - The two are NOT equivalent
  - JFLAP comes to the rescue yet again!
  - Questions?