Foundations of Computing

Lecture 8

Arkady Yerukhimovich

February 8, 2024

Outline

Lecture 7 Review

- Proving languages not regular
  - Using the pumping lemma
  - Using closure properties

Lecture 7 Review

- Proving languages not regular
    - Using the pumping lemma
    - Using closure properties

Today

Going beyond regular languages.

How Can We Recognize Non-Regular Languages?

Let $L = \{0^n 1^n | n \geq 0\}$

Question

How can we build a machine to recognize this language?

How Can We Recognize Non-Regular Languages?

Let $L = \{0^n 1^n | n \geq 0\}$

Question

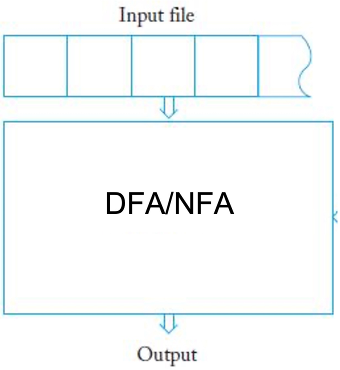
How can we build a machine to recognize this language?

Answer

Add some form of (unbounded) memory to the machine

Outline

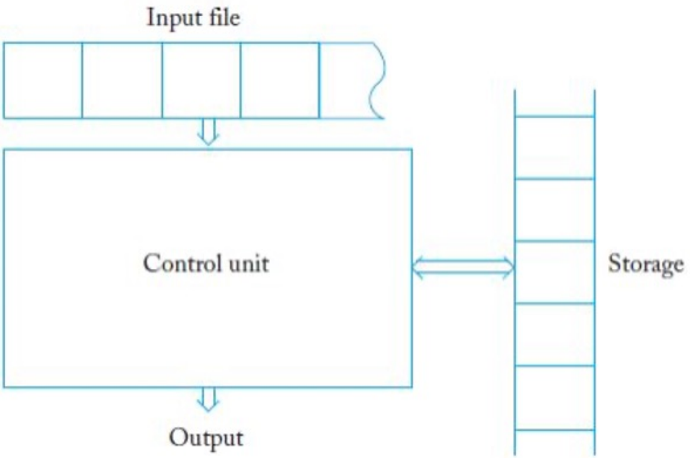
Let's Add Some Storage



Recall:

- An NFA/DFA has no external storage
- Only memory must be encoded in the finite number of states
- Can only recognize regular languages

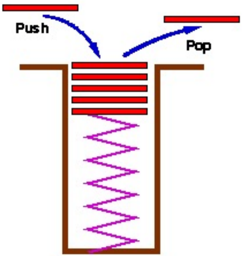
Let's Add Some Storage



Question

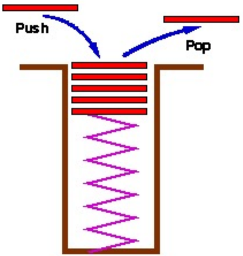
What kind of storage should we add?

A Stack



Let's add a Stack for storage

A Stack

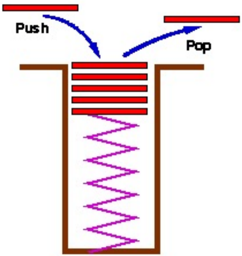


Let's add a Stack for storage

A stack has the following operations:

- push value - push a value onto the top of the stack

A Stack

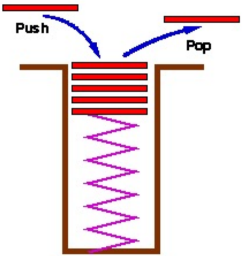


Let's add a Stack for storage

A stack has the following operations:

- push value - push a value onto the top of the stack
- pop value - pop the top item off the stack

A Stack

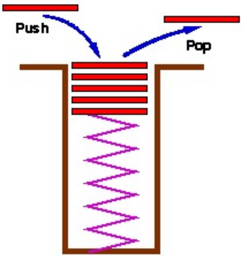


Let's add a Stack for storage

A stack has the following operations:

- push value - push a value onto the top of the stack
- pop value - pop the top item off the stack
- do nothing - denoted as ϵ

A Stack



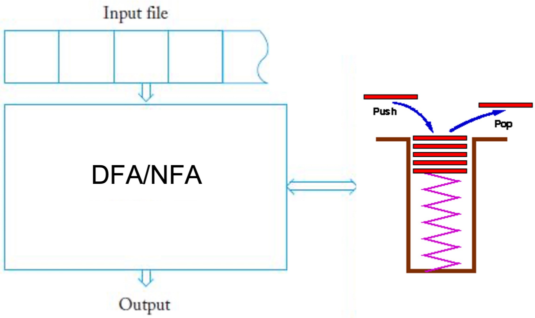
Let's add a Stack for storage

A stack has the following operations:

- push value - push a value onto the top of the stack
- pop value - pop the top item off the stack
- do nothing - denoted as ϵ

A stack is a Last-In First-Out (LIFO) data structure,
that can hold an infinite amount of information (infinite depth)

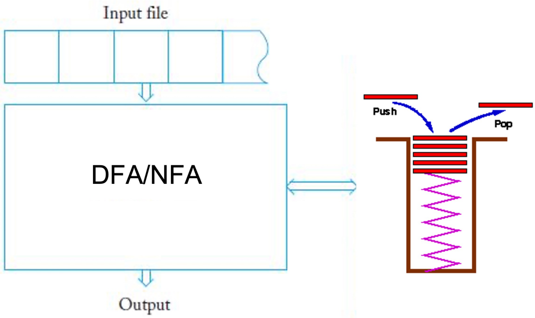
Pushdown Automata (PDA)



A PDA consists of:

- An NFA for a control unit
- A Stack for storage

Pushdown Automata (PDA)



A PDA consists of:

- An NFA for a control unit
- A Stack for storage

Question

Is this any more powerful than an NFA?

Computing With a PDA

Computing with a PDA

At each step, a PDA can do the following

Computing With a PDA

Computing with a PDA

At each step, a PDA can do the following

1. Read a symbol from the input tape

Computing With a PDA

Computing with a PDA

At each step, a PDA can do the following

1. Read a symbol from the input tape
2. Optionally, pop a value from the Stack

Computing With a PDA

Computing with a PDA

At each step, a PDA can do the following

1. Read a symbol from the input tape
2. Optionally, pop a value from the Stack
3. Use the input symbol and the Stack symbol to choose a next state

Computing With a PDA

Computing with a PDA

At each step, a PDA can do the following

1. Read a symbol from the input tape
2. Optionally, pop a value from the Stack
3. Use the input symbol and the Stack symbol to choose a next state
4. Optionally, push a value onto the Stack

Computing With a PDA

Computing with a PDA

At each step, a PDA can do the following

1. Read a symbol from the input tape
2. Optionally, pop a value from the Stack
3. Use the input symbol and the Stack symbol to choose a next state
4. Optionally, push a value onto the Stack

A PDA M accepts a string w if the NFA in the control stops in an accept state once all the input has been processed

Computing With a PDA

Computing with a PDA

At each step, a PDA can do the following

1. Read a symbol from the input tape
2. Optionally, pop a value from the Stack
3. Use the input symbol and the Stack symbol to choose a next state
4. Optionally, push a value onto the Stack

A PDA M accepts a string w if the NFA in the control stops in an accept state once all the input has been processed

Observations:

Computing With a PDA

Computing with a PDA

At each step, a PDA can do the following

1. Read a symbol from the input tape
2. Optionally, pop a value from the Stack
3. Use the input symbol and the Stack symbol to choose a next state
4. Optionally, push a value onto the Stack

A PDA M accepts a string w if the NFA in the control stops in an accept state once all the input has been processed

Observations:

- Since the control is an NFA, ϵ transitions are allowed

Computing With a PDA

Computing with a PDA

At each step, a PDA can do the following

1. Read a symbol from the input tape
2. Optionally, pop a value from the Stack
3. Use the input symbol and the Stack symbol to choose a next state
4. Optionally, push a value onto the Stack

A PDA M accepts a string w if the NFA in the control stops in an accept state once all the input has been processed

Observations:

- Since the control is an NFA, ϵ transitions are allowed
- A PDA may choose not to touch the stack in a particular step

Computing With a PDA

Computing with a PDA

At each step, a PDA can do the following

1. Read a symbol from the input tape
2. Optionally, pop a value from the Stack
3. Use the input symbol and the Stack symbol to choose a next state
4. Optionally, push a value onto the Stack

A PDA M accepts a string w if the NFA in the control stops in an accept state once all the input has been processed

Observations:

- Since the control is an NFA, ϵ transitions are allowed
- A PDA may choose not to touch the stack in a particular step
- Unlike the case for DFA/NFA, deterministic PDA's are not equal to non-deterministic ones. We will only study non-deterministic PDAs.

An Example PDA

A PDA for $L = \{0^n 1^n | n \geq 0\}$

An Example PDA

A PDA for $L = \{0^n 1^n | n \geq 0\}$

Consider the following PDA "Algorithm"

An Example PDA

A PDA for $L = \{0^n 1^n | n \geq 0\}$

Consider the following PDA "Algorithm"

1. Read a symbol from the input

An Example PDA

A PDA for $L = \{0^n 1^n | n \geq 0\}$

Consider the following PDA "Algorithm"

1. Read a symbol from the input
2. If it is a 0 and I have not seen any 1s, then push a 0 onto the stack

An Example PDA

A PDA for $L = \{0^n 1^n | n \geq 0\}$

Consider the following PDA "Algorithm"

1. Read a symbol from the input
2. If it is a 0 and I have not seen any 1s, then push a 0 onto the stack
3. If it is a 1, pop a value (a 0) from the stack

An Example PDA

A PDA for $L = \{0^n 1^n | n \geq 0\}$

Consider the following PDA "Algorithm"

1. Read a symbol from the input
2. If it is a 0 and I have not seen any 1s, then push a 0 onto the stack
3. If it is a 1, pop a value (a 0) from the stack
4. Accept if and only if the stack becomes empty when we read the last character

An Example PDA

A PDA for $L = \{0^n 1^n | n \geq 0\}$

Consider the following PDA "Algorithm"

1. Read a symbol from the input
2. If it is a 0 and I have not seen any 1s, then push a 0 onto the stack
3. If it is a 1, pop a value (a 0) from the stack
4. Accept if and only if the stack becomes empty when we read the last character
5. Reject if any of the following happen:
    - the stack becomes empty and the input is not done or

An Example PDA

A PDA for $L = \{0^n 1^n | n \geq 0\}$

Consider the following PDA "Algorithm"

1. Read a symbol from the input
2. If it is a 0 and I have not seen any 1s, then push a 0 onto the stack
3. If it is a 1, pop a value (a 0) from the stack
4. Accept if and only if the stack becomes empty when we read the last character
5. Reject if any of the following happen:
   - the stack becomes empty and the input is not done or
   - there are still 0s left on the stack when the last input is read or

An Example PDA

A PDA for $L = \{0^n 1^n | n \geq 0\}$

Consider the following PDA "Algorithm"

1. Read a symbol from the input
2. If it is a 0 and I have not seen any 1s, then push a 0 onto the stack
3. If it is a 1, pop a value (a 0) from the stack
4. Accept if and only if the stack becomes empty when we read the last character
5. Reject if any of the following happen:
    - the stack becomes empty and the input is not done or
    - there are still 0s left on the stack when the last input is read or
    - there are any 0s after the first 1

An Example PDA

A PDA for $L = \{0^n 1^n | n \geq 0\}$

Consider the following PDA "Algorithm"

1. Read a symbol from the input
2. If it is a 0 and I have not seen any 1s, then push a 0 onto the stack
3. If it is a 1, pop a value (a 0) from the stack
4. Accept if and only if the stack becomes empty when we read the last character
5. Reject if any of the following happen:
   - the stack becomes empty and the input is not done or
   - there are still 0s left on the stack when the last input is read or
   - there are any 0s after the first 1

Outline

Formal Definition of PDAs

A PDA M is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where

- Q – set of states of the NFA
- Σ – input alphabet
- Γ – Stack alphabet
- $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \rightarrow P(Q \times \Gamma_\epsilon)$ – transition function
- $q_0 \in Q$ – start state
- $F \subseteq Q$ – accept states

Recall that $P(Q \times \Gamma_\epsilon)$ is the power set of the set of pairs $\{(q \in Q, a \in \Gamma_\epsilon)\}$

Computing with a PDA – Formal Notation

A PDA M accepts a string $w = w_1 w_2 \cdots w_m$ with $w_i \in \Sigma_\epsilon$ if there exist

- A sequence of states $q_0, q_1, \ldots q_m \in Q$, and
- A sequence of strings $s_0, s_1, \ldots, s_m \in \Gamma^*$

that satisfy the following three conditions:

Computing with a PDA – Formal Notation

A PDA M accepts a string $w = w_1 w_2 \cdots w_m$ with $w_i \in \Sigma_\epsilon$ if there exist

- A sequence of states $q_0, q_1, \ldots q_m \in Q$, and
- A sequence of strings $s_0, s_1, \ldots, s_m \in \Gamma^*$

that satisfy the following three conditions:

1. q_0 is the start state, and $s_0 = \epsilon$
   M starts in the start state with an empty stack

Computing with a PDA – Formal Notation

A PDA M accepts a string $w = w_1 w_2 \cdots w_m$ with $w_i \in \Sigma_\epsilon$ if there exist

- A sequence of states $q_0, q_1, \ldots q_m \in Q$, and
- A sequence of strings $s_0, s_1, \ldots, s_m \in \Gamma^*$

that satisfy the following three conditions:

1. q_0 is the start state, and $s_0 = \epsilon$
   M starts in the start state with an empty stack
2. For $i = 0, \ldots, m-1$, $(q_{i+1}, b) \in \delta(q_i, w_{i+1}, a)$ where $s_i = at$ and $s_{i+1} = bt$ for some $a, b \in \Gamma_\epsilon$ and $t \in \Gamma^*$
   there is a transition in δ s.t. M reads symbol w_{i+1} from the input, pops a from the stack, pushes b back on the stack and moves from q_i to q_{i+1}

Computing with a PDA – Formal Notation

A PDA M accepts a string $w = w_1 w_2 \cdots w_m$ with $w_i \in \Sigma_\epsilon$ if there exist

- A sequence of states $q_0, q_1, \ldots q_m \in Q$, and
- A sequence of strings $s_0, s_1, \ldots, s_m \in \Gamma^*$

that satisfy the following three conditions:

1. q_0 is the start state, and $s_0 = \epsilon$
   M starts in the start state with an empty stack
2. For $i = 0, \ldots, m-1$, $(q_{i+1}, b) \in \delta(q_i, w_{i+1}, a)$ where $s_i = at$ and $s_{i+1} = bt$ for some $a, b \in \Gamma_\epsilon$ and $t \in \Gamma^*$
   there is a transition in δ s.t. M reads symbol w_{i+1} from the input, pops a from the stack, pushes b back on the stack and moves from q_i to q_{i+1}
3. $r_m \in F$ – stop in an accept state

Back to Our Example

Recall the PDA we described before:

- On input 0, push a 0 on the stack
- On input 1, pop a value from the stack
- If all 0s come before all 1s and the stack is empty when run out of inputs, accept

Back to Our Example

Recall the PDA we described before:

- On input 0, push a 0 on the stack
- On input 1, pop a value from the stack
- If all 0s come before all 1s and the stack is empty when run out of inputs, accept

Let's build a PDA for this algorithm:

Back to Our Example

Recall the PDA we described before:

- On input 0, push a 0 on the stack
- On input 1, pop a value from the stack
- If all 0s come before all 1s and the stack is empty when run out of inputs, accept

Let's build a PDA for this algorithm:

- $Q = \{q_0, q_1, q_2, q_3\}$
  - q_0 – start state
  - q_1 – seen only 0s
  - q_2 – seen 0s followed by 1s
  - q_3 – accept state

Back to Our Example

Recall the PDA we described before:

- On input 0, push a 0 on the stack
- On input 1, pop a value from the stack
- If all 0s come before all 1s and the stack is empty when run out of inputs, accept

Let's build a PDA for this algorithm:

- $Q = \{q_0, q_1, q_2, q_3\}$
  - q_0 – start state
  - q_1 – seen only 0s
  - q_2 – seen 0s followed by 1s
  - q_3 – accept state
- $\Sigma = \{0, 1\}$

Back to Our Example

Recall the PDA we described before:

- On input 0, push a 0 on the stack
- On input 1, pop a value from the stack
- If all 0s come before all 1s and the stack is empty when run out of inputs, accept

Let's build a PDA for this algorithm:

- $Q = \{q_0, q_1, q_2, q_3\}$
  - q_0 – start state
  - q_1 – seen only 0s
  - q_2 – seen 0s followed by 1s
  - q_3 – accept state
- $\Sigma = \{0, 1\}$
- $\Gamma = \{0, \$\}$ – \$ is a special symbol to indicate the stack is empty

Back to Our Example

Recall the PDA we described before:

- On input 0, push a 0 on the stack
- On input 1, pop a value from the stack
- If all 0s come before all 1s and the stack is empty when run out of inputs, accept

Let's build a PDA for this algorithm:

- $Q = \{q_0, q_1, q_2, q_3\}$
  - q_0 – start state
  - q_1 – seen only 0s
  - q_2 – seen 0s followed by 1s
  - q_3 – accept state
- $\Sigma = \{0, 1\}$
- $\Gamma = \{0, \$\}$ – \$ is a special symbol to indicate the stack is empty
- $q_0 = q_0$

Back to Our Example

Recall the PDA we described before:

- On input 0, push a 0 on the stack
- On input 1, pop a value from the stack
- If all 0s come before all 1s and the stack is empty when run out of inputs, accept

Let's build a PDA for this algorithm:

- $Q = \{q_0, q_1, q_2, q_3\}$
  - q_0 – start state
  - q_1 – seen only 0s
  - q_2 – seen 0s followed by 1s
  - q_3 – accept state
- $\Sigma = \{0, 1\}$
- $\Gamma = \{0, \$\}$ – \$ is a special symbol to indicate the stack is empty
- $q_0 = q_0$
- $F = \{q_3\}$

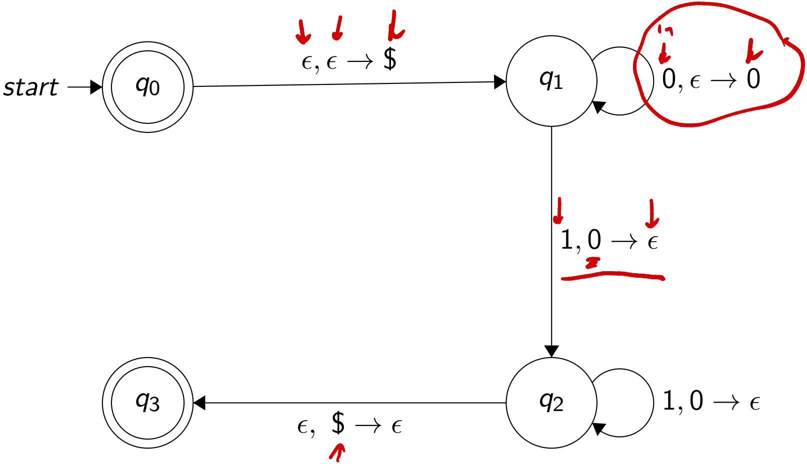
Transition Function

Input:	0			1			ϵ		
Stack:	0	\$	ϵ	0	\$	ϵ	0	\$	ϵ
q_0									$\{(q_1, \$)\}$
q_1			$\{(q_1, 0)\}$	$\{(q_2, \epsilon)\}$					
q_2				$\{(q_2, \epsilon)\}$				$\{(q_3, \epsilon)\}$	
q_3									

Table: Transition Function δ

Empty cells correspond to output of $\emptyset$

Example PDA as a Graph



Exercise – Work in Groups

Show a PDA that recognizes the language

$$L = \{w \mid w \text{ has an equal number of 0s and 1s}\}$$

1. Describe a PDA algorithm for this language
2. Write the states and transition function
3. Draw the PDA graph

Exercise – Work in Groups

Show a PDA that recognizes the language

$$L = \{w \mid w \text{ has an equal number of 0s and 1s}\}$$

1. Describe a PDA algorithm for this language
2. Write the states and transition function
3. Draw the PDA graph

Solution:

1. Push \$ on the stack
2. If input is 0, pop value from the stack
   - If it's a 0 or \$ push it back on the stack and push another 0 on top
   - If it's a 1 pop it off the stack
3. If input is 1, pop value from the stack
   - If it's a 1 or \$ push it back and push another 1 on top
   - If it's a 0 pop it off the stack
4. When the input is done, if \$ is top of the stack, accept

Exercise – Work in Groups