Formal Language & Automata Theory

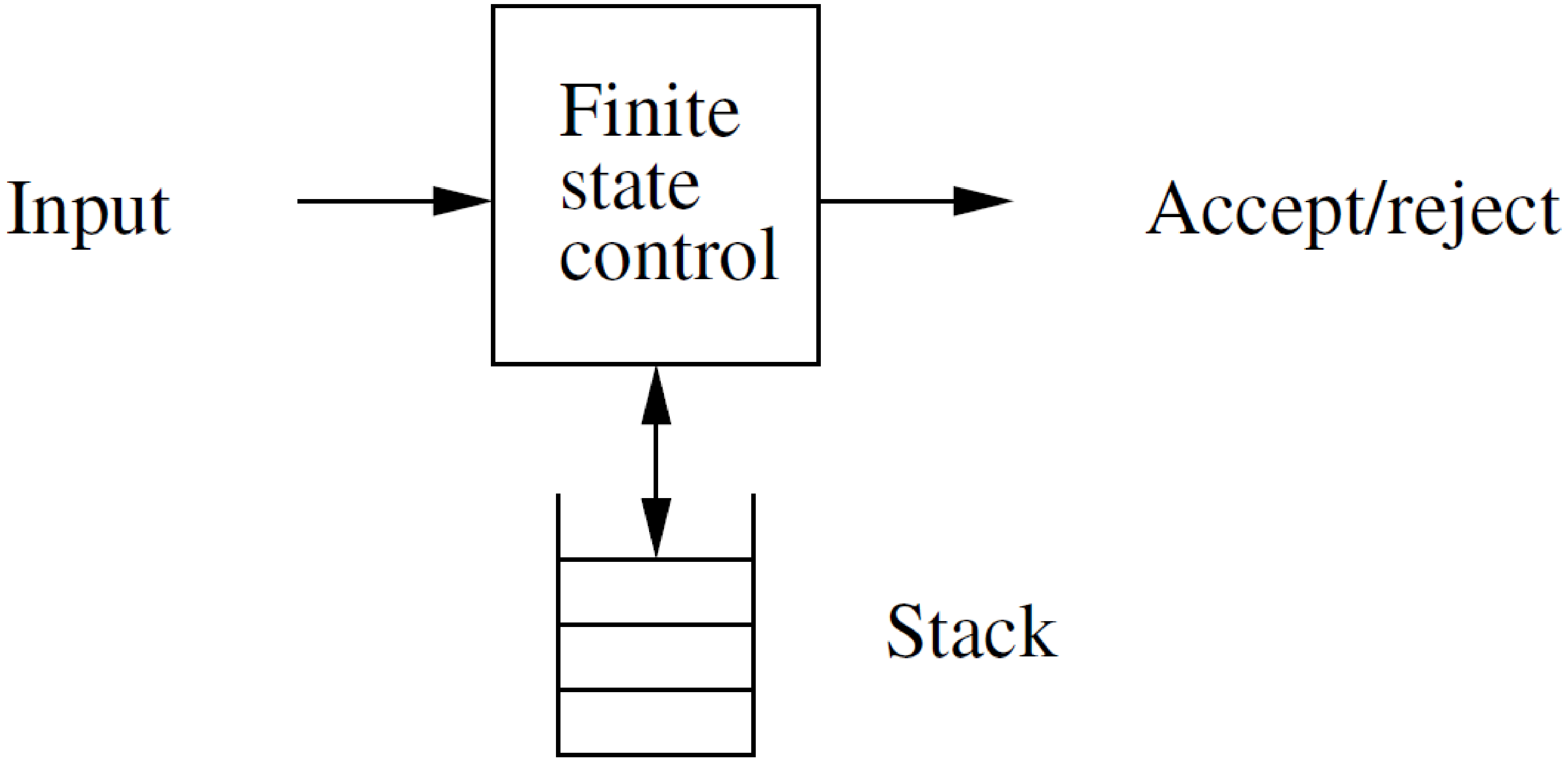
Prof. Sankhadeep Chatterjee

Push Down Automata (PDA)

- Informally, an ε-NFA with the additional power that it can manipulate a stack.

- Its moves are determined by:
  1. The current state (of its "NFA"),
  2. The current input symbol (or ε), and
  3. The current symbol on top of its stack.

Push Down Automata (PDA)



Push Down Automata (PDA)

- Being nondeterministic, the PDA can have a choice of next moves.
- In each choice, the PDA can:
  1. Change state
  2. Replace the top symbol on the stack by a sequence of zero or more symbols.
     - Zero symbols = “pop.”
     - Many symbols = sequence of “pushes.”

Push Down Automata (PDA)

- A PDA is described by:
  1. A finite set of states (Q).
  2. An input alphabet (Σ).
  3. A stack alphabet (Γ).
  4. A transition function (δ).
  5. A start state ($q_0 \in Q$).
  6. A start symbol ($Z \in \Gamma$).
  7. A set of final states ($F \subseteq Q$).

- Some Conventions

1. a, b, … are input symbols.
   - But sometimes we allow ε as a possible value.
2. …, X, Y, Z are stack symbols.
3. …, w, x, y, z are strings of input symbols.
4. α, β,… are strings of stack symbols.

Push Down Automata (PDA)

- Takes three arguments:
  1. A state, in Q.
  2. An input, which is either a symbol in Σ or ε.
  3. A stack symbol in Γ.
- $\delta(q, a, X)$ is a set of zero or more actions of the form (p, α).
  - p is a state; α is a string of stack symbols.
- If $\delta(q, a, X)$ contains (p, α) among its actions, then one thing the PDA can do in state q, with a at the front of the input, and X on top of the stack is:
  1. Change the state to p.
  2. Remove a from the front of the input (but a may be ε).
  3. Replace X on the top of the stack by α.

Example: PDA

- Design a PDA to accept $\{0^n 1^n \mid n \geq 1\}$.
- The states:
  - q = start state. We are in state q if we have seen only 0's so far.
  - p = we've seen at least one 1 and may now proceed only if the inputs are 1's.
  - f = final state; accept.
- The stack symbols:
  - Z = start symbol. Also marks the bottom of the stack, so we know when we have counted the same number of 1's as 0's.
  - X = marker, used to count the number of 0's seen on the input.

Example: PDA

- The transitions:
  - $\delta(q, 0, Z) = \{(q, XZ)\}$.
  - $\delta(q, 0, X) = \{(q, XX)\}$.
  - $\delta(q, 1, X) = \{(p, \varepsilon)\}$.
  - $\delta(p, 1, X) = \{(p, \varepsilon)\}$.
  - $\delta(p, \varepsilon, Z) = \{(f, Z)\}$.

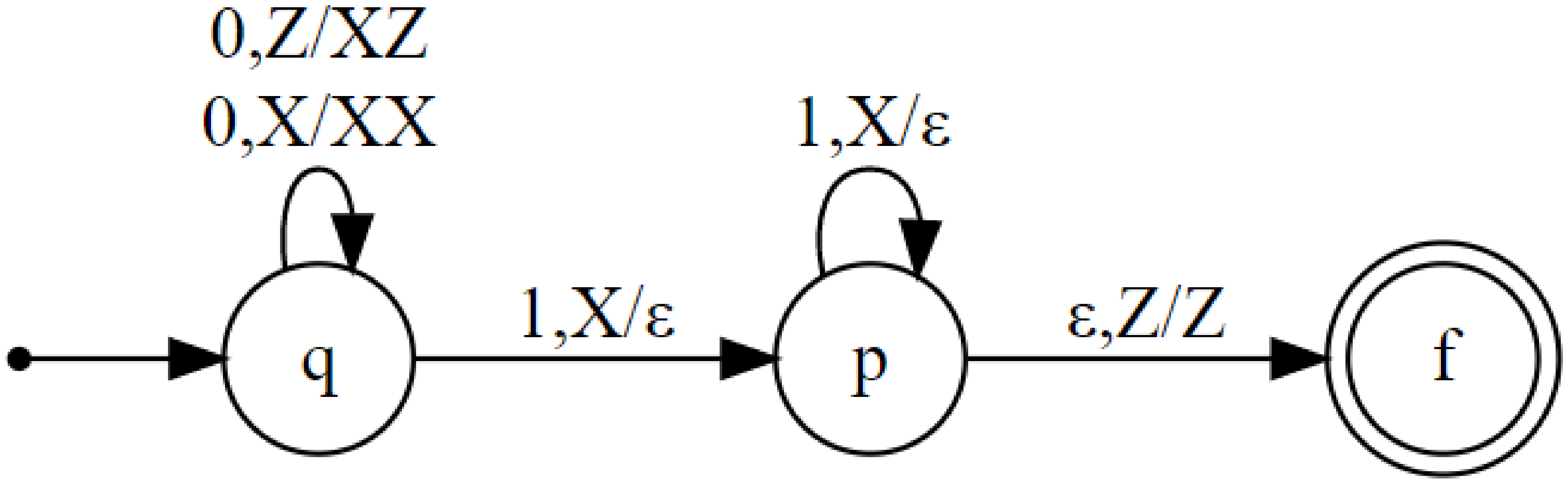
These two rules cause one X to be pushed onto the stack for each 0 read from the input.

When we see a 1, go to state p and pop one X.

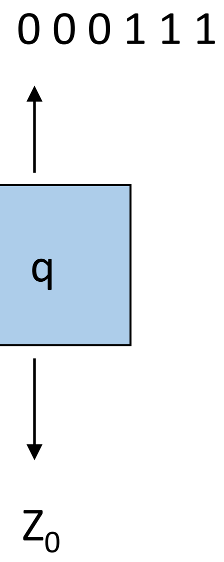
Pop one X per 1.

Accept at bottom.

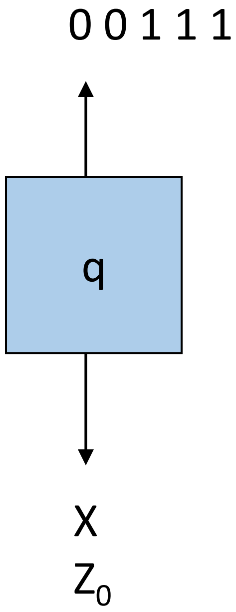
Example: PDA (Transition Diagram)



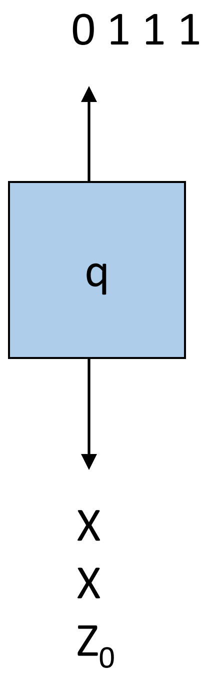
Actions of the Example PDA



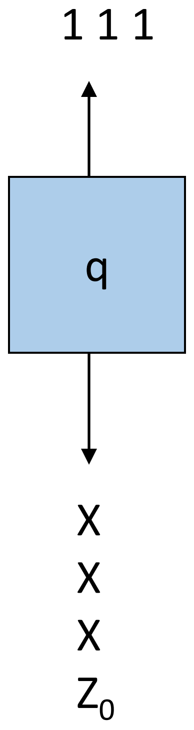
Actions of the Example PDA



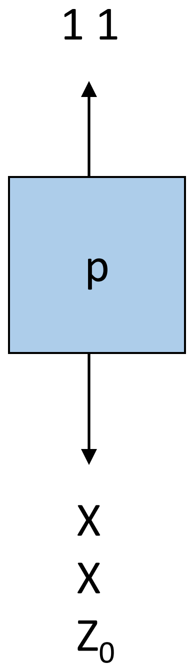
Actions of the Example PDA



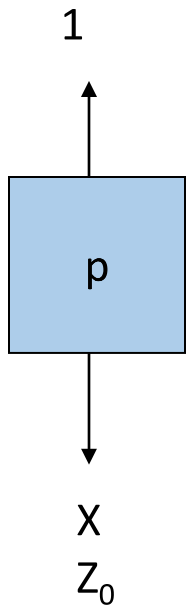
Actions of the Example PDA



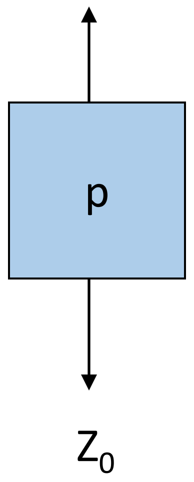
Actions of the Example PDA



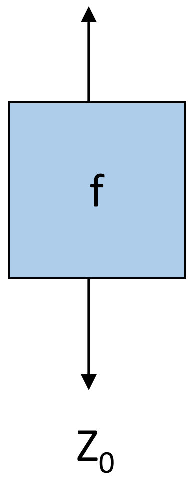
Actions of the Example PDA



Actions of the Example PDA



Actions of the Example PDA



Instantaneous Descriptions

- We can formalize the pictures just seen with an instantaneous description (ID).
- A ID is a triple (q, w, α), where:
  1. q is the current state.
  2. w is the remaining input.
  3. α is the stack contents, top at the left.

The "Goes-To" Relation

- To say that ID I can become ID J in one move of the PDA, we write $I \vdash J$.
- Formally, (q, aw, Xα) ⊢ (p, w, $\beta\alpha$) for any w and α, if δ(q, a, X) contains (p, β).
- Extend ⊢ to ⊢*, meaning "zero or more moves," by:
  - Basis: $I \vdash^* I$.
  - Induction: If $I \vdash^* J$ and $J \vdash K$, then $I \vdash^* K$.

Using the previous example PDA, we can describe the sequence of moves by:

(q, 000111, Z_0)

⊢ (q, 00111, X Z_0)

⊢ (q, 0111, XX Z_0)

⊢ (q, 111, XXX Z_0)

⊢(p, 11, XX Z_0)

⊢ (p, 1, X Z_0)

⊢ (p, ε, Z_0)

⊢ (f, ε, Z_0)

Acceptance of String by PDA

Acceptance by ***final state***:

String w is said to be accepted by final state if $(q_0, w, Z_0) \vdash^* (f, \varepsilon, \alpha)$ for final state f and any α.

Acceptance by ***empty stack***:

String w is said to be accepted by empty stack if $(q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon)$ for any state q.

CFG to PDA

Let $G = (V, T, Q, S)$ be a CFG. Construct the PDA P that accepts L(G) by empty stack as follows:

$$P = (\{q\}, T, V \cup T, \delta, q, S)$$

Where transition function δ is defined as:

1. For each variable A

$$\delta(q, \epsilon, A) = \{(q, \beta) | A \to \beta \text{ is a production in G}\}$$

2. For each terminal a

$$\delta(q, a, a) = \{(q, \epsilon)\}$$

Example: CFG to PDA

Grammar:

$$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$$
$$E \to I \mid E * E \mid E + E \mid (E)$$

Terminals (T) = $\{a, b, 0,1, +,*, (,)\}$

Stack Symbols ($V \cup T$) = $\{I, E, a, b, 0,1, +,*, (,)\}$

Example: CFG to PDA

The transition function (δ) is defined as follows:

From Rule 1 (variables) we get

1. $\delta(q, \epsilon, I) = \{(q, a), (q, b), (q, Ia), (q, Ib), (q, I0), (q, I1)\}$
2. $\delta(q, \epsilon, E) = \{(q, I), (q, E + E), (q, E * E), (q, (E))\}$

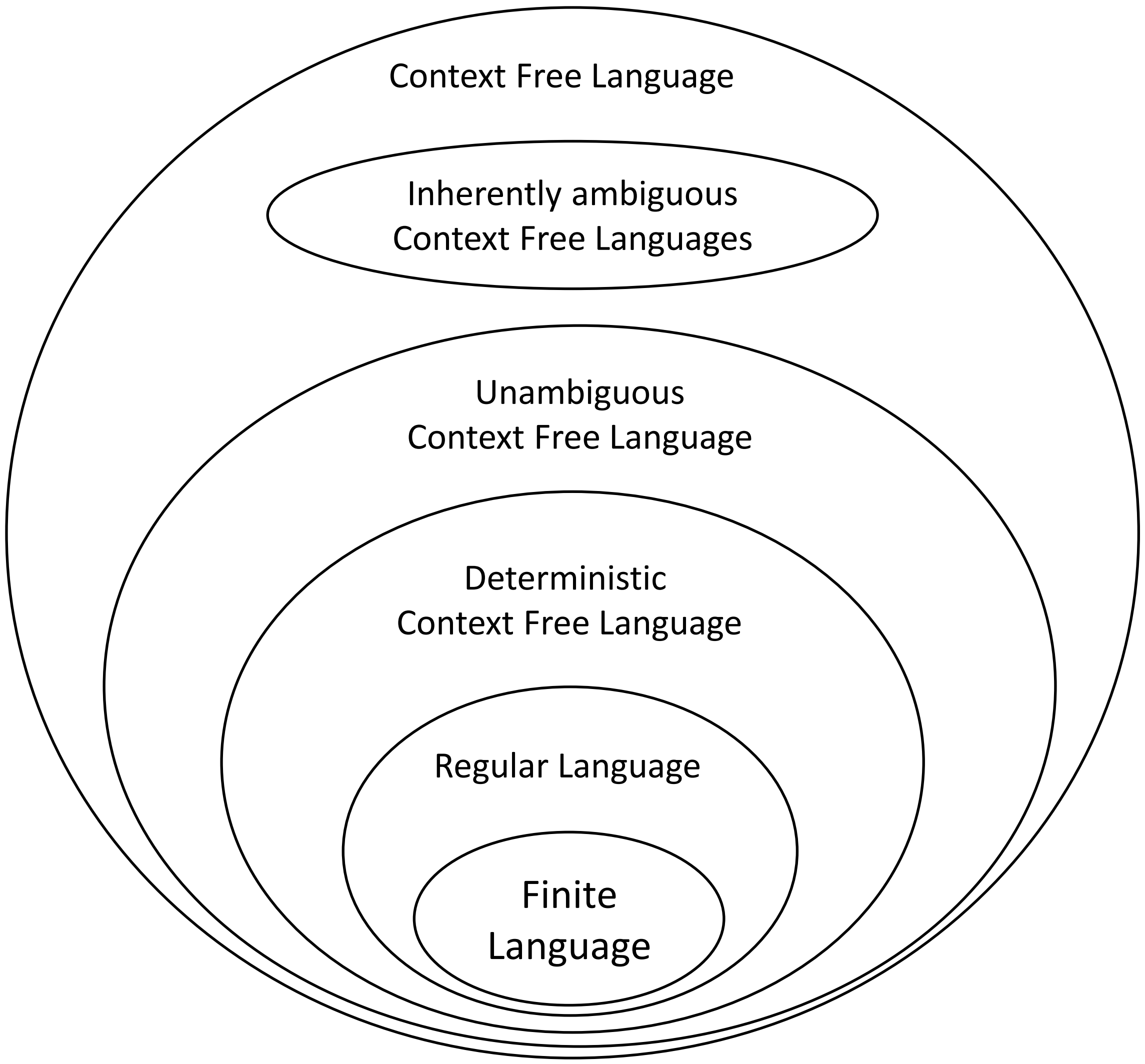
From Rule 2 (terminals) we get

1. $\delta(q, a, a) = (q, \epsilon)$, $\delta(q, b, b) = (q, \epsilon)$, $\delta(q, 0,0) = (q, \epsilon)$
   $\delta(q, 1,1) = (q, \epsilon)$, $\delta(q, (, () = (q, \epsilon)$, $\delta(q,),)) = (q, \epsilon)$,
   $\delta(q, +, +) = (q, \epsilon)$, $\delta(q,*,*) = (q, \epsilon)$

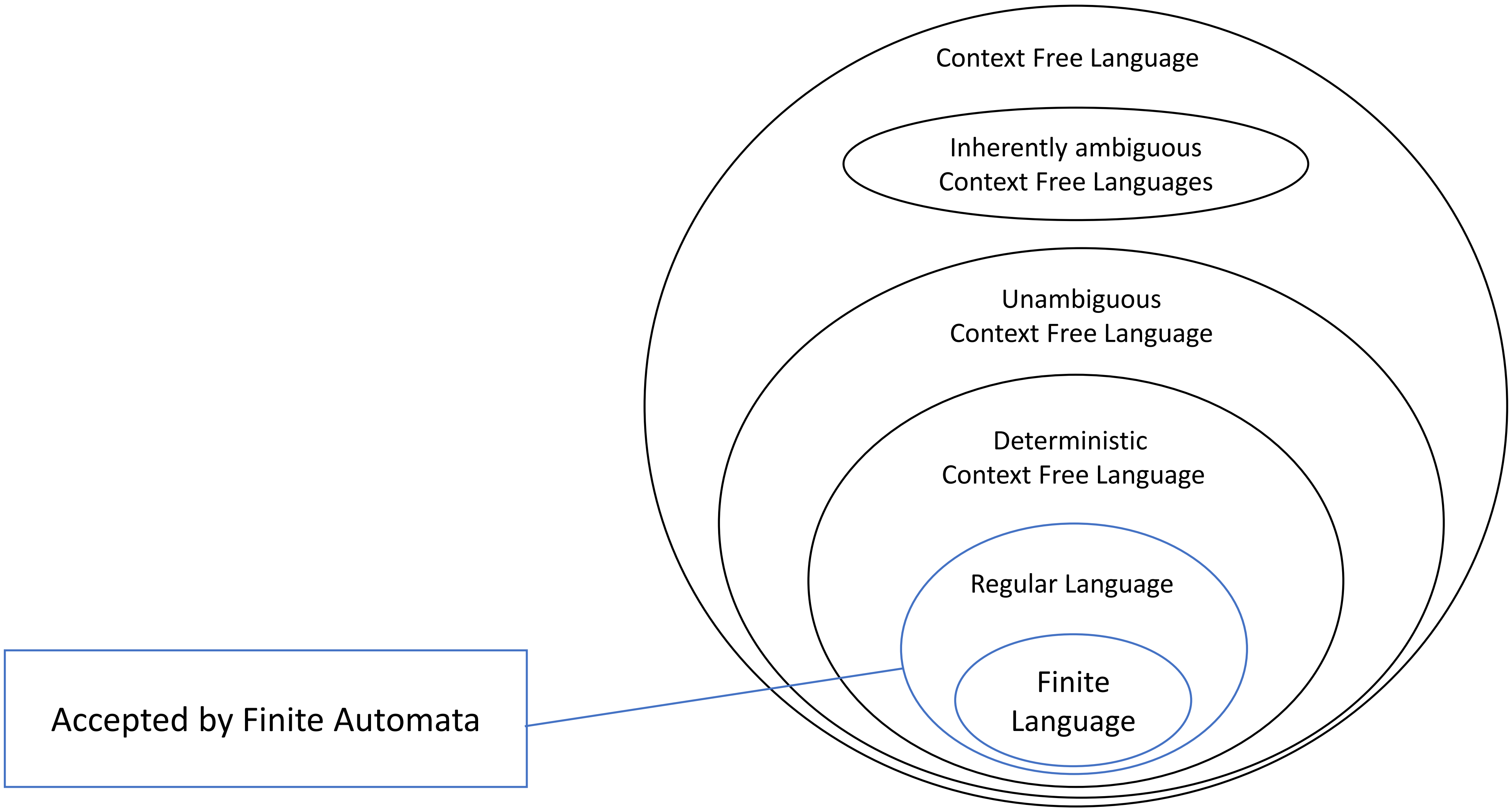
Deterministic PDA

- To be deterministic, there must be at most one choice of move for any state q, input symbol *a*, and stack symbol X.
- In addition, there must not be a choice between using input ε or real input.
- Formally, $\delta(q, a, X)$ and $\delta(q, \varepsilon, X)$ cannot both be nonempty.

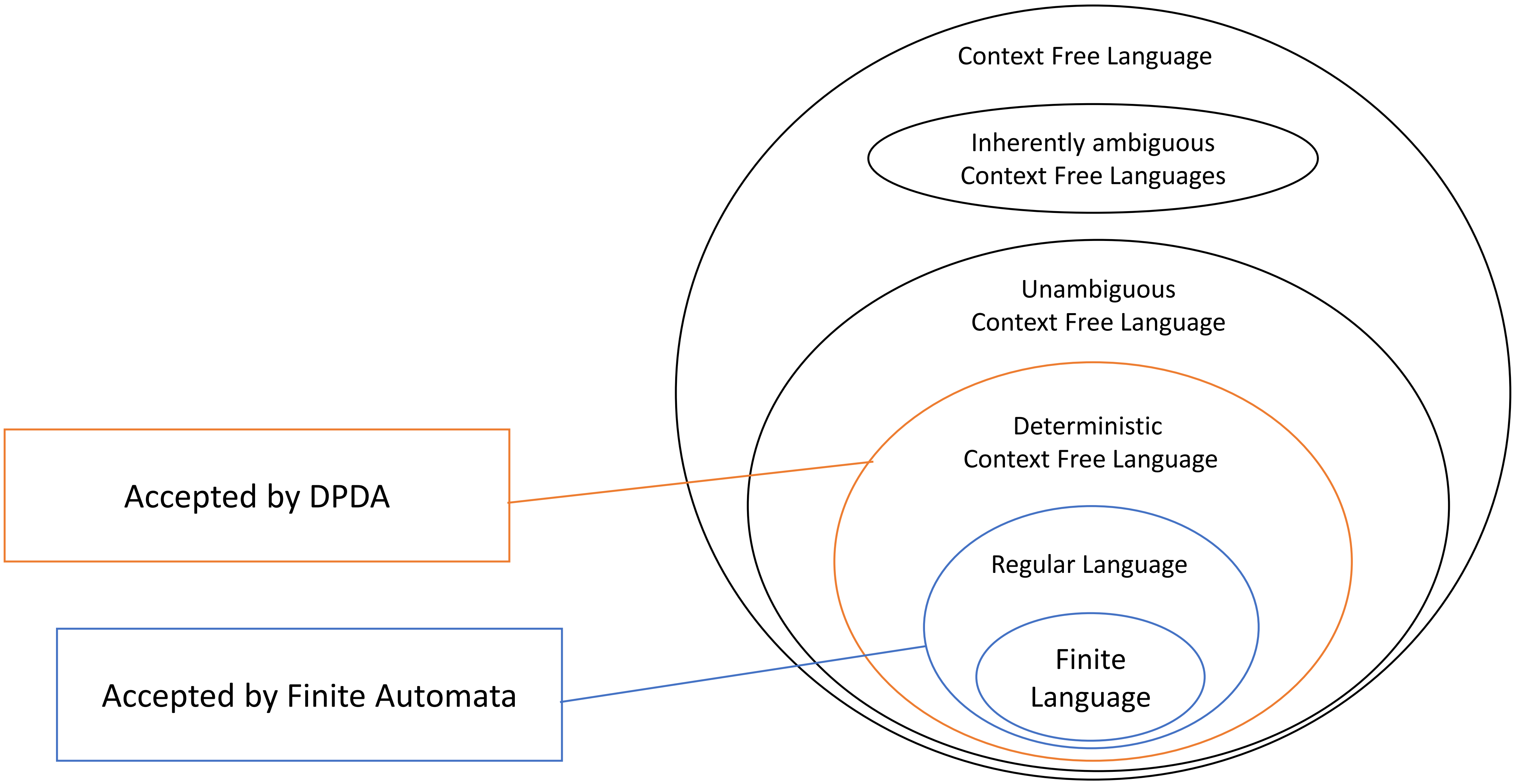
Formal Language Classes



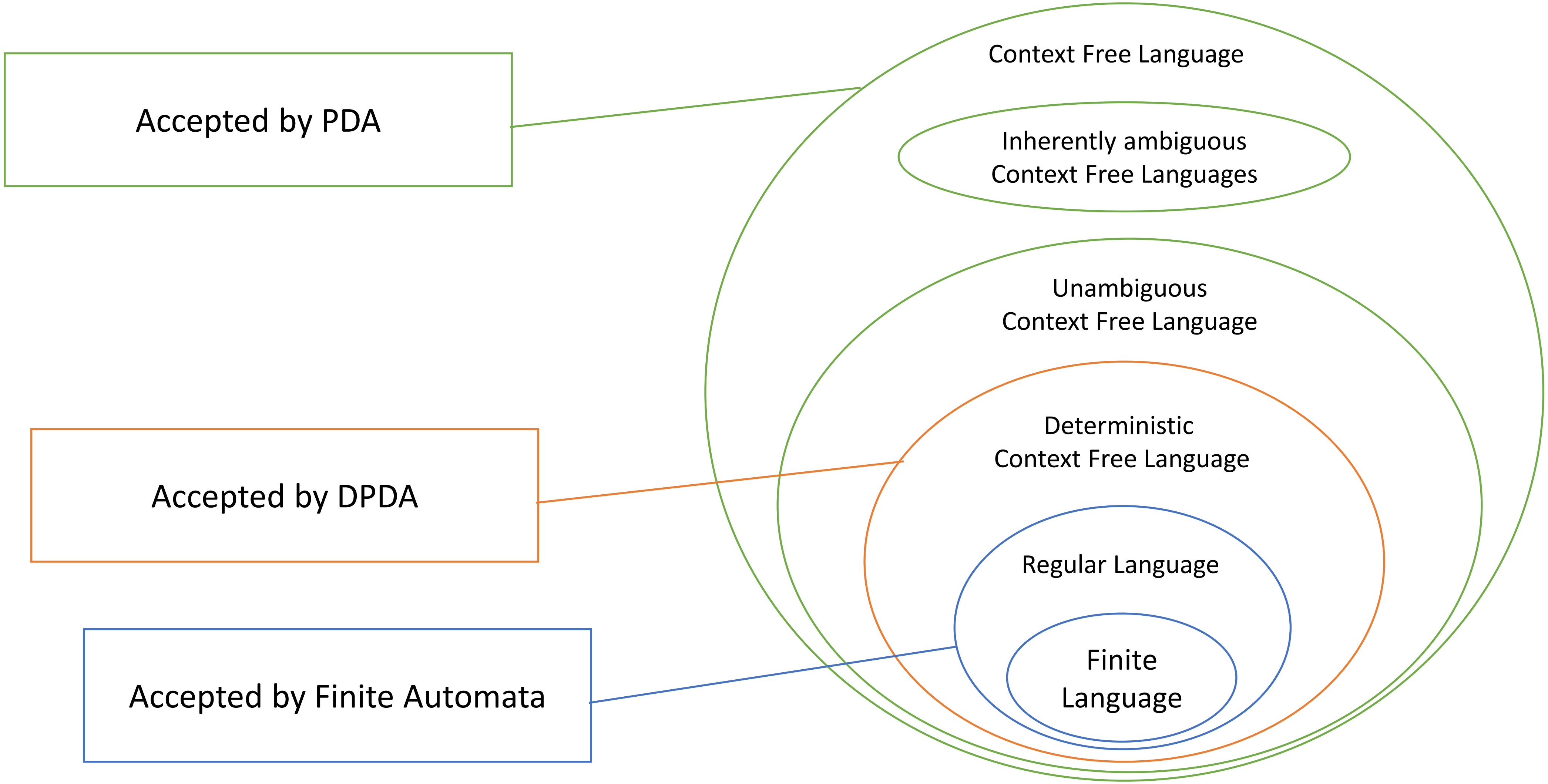
Formal Language Classes



Formal Language Classes



Formal Language Classes



Pumping Lemma for CFL

For every context-free language L

There is an integer n, such that

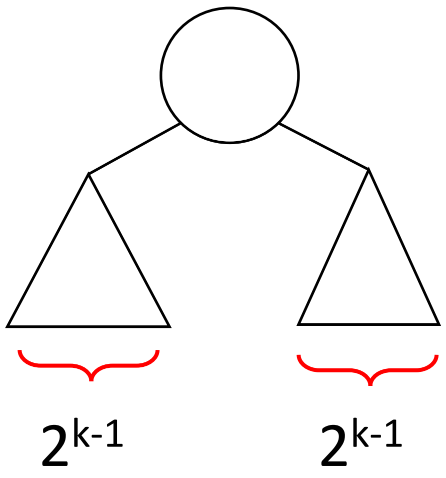
For every string z in L of length $\geq$ n

There exists z = uvwxy such that:

1. $|vwx| \leq n$.
2. $|vx| > 0$.
3. For all $i \geq 0$, uv^iwx^iy is in L.

Yield of a Parse Tree of CNF grammar

- Length of path in parse tree is number of edges along a path
- Parse tree of a CNF grammar with maximum path length ‘n’ can yield a string of length <= 2^{n-1}
- Basis: if n = 1, then the tree has root node and a terminal node only. The string thus generated is of length $2^{1-1} = 1$
- Induction: If the hypothesis is true for any ‘k’ (>= 1), then for maximum path length k+1, the tree looks like the following:

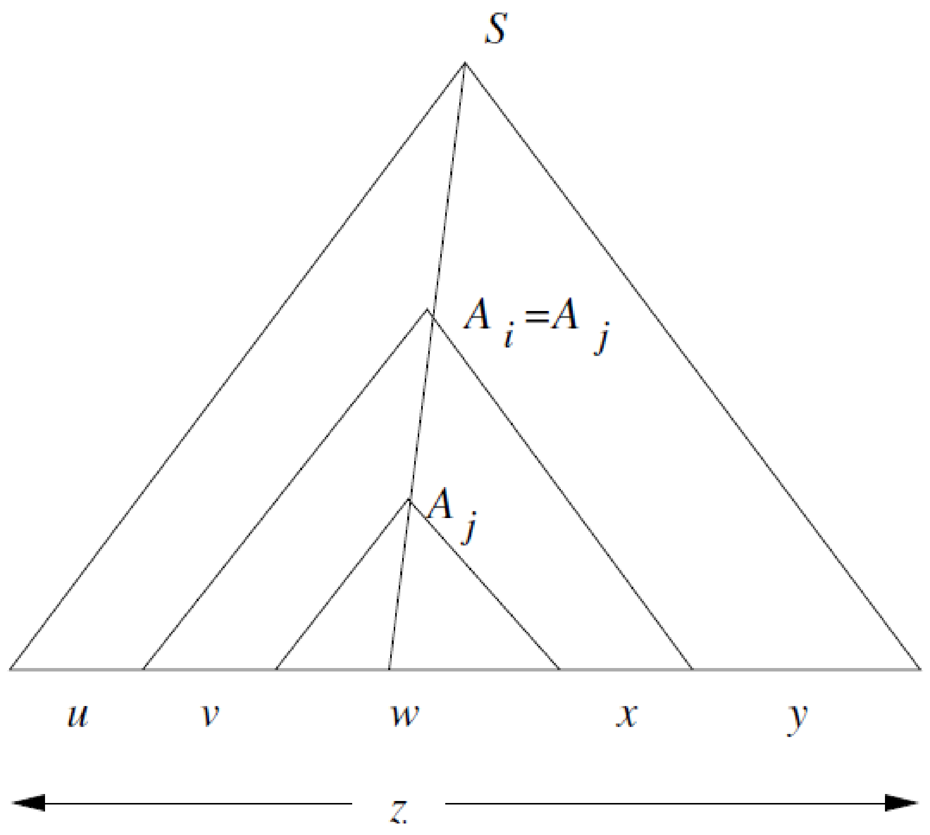


- Both left and right sub-tree can have maximum path length of k i.e maximum yield 2^{k-1}
- Thus total length of yield from both sub tree is

$2^{k-1} + 2^{k-1} = 2^* \, 2^{k-1} = 2^{(k+1)-1}$ (Proved)

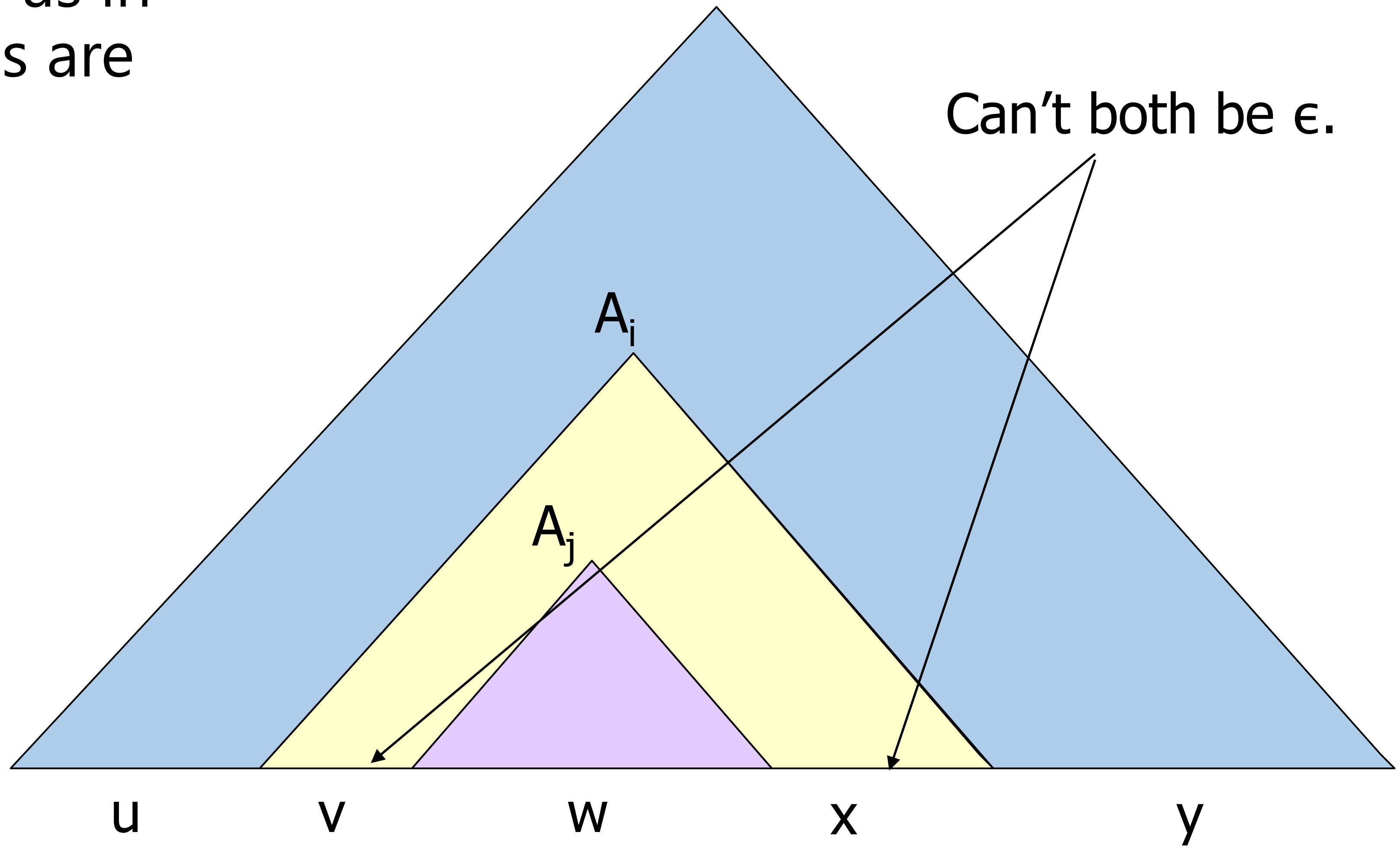
Proof: Pumping Lemma for CFL

- Start with a CNF grammar for L – {ε}.
- Let the grammar have m variables. Pick $n = 2^m$. Let $|z| >= n$.
- To generate a string of length 2^m the path length is m+1.
- That indicates in that path there are m+2 many nodes
- Out of which first (1,2,…m+1) are variables and (m+2) th node is terminal
- But, we have only 'm' variables in the grammar. Hence, one variable must be repeated in that path.



Proof: Pumping Lemma for CFL

- v, x cant both be null as in CNF Unit productions are not allowed



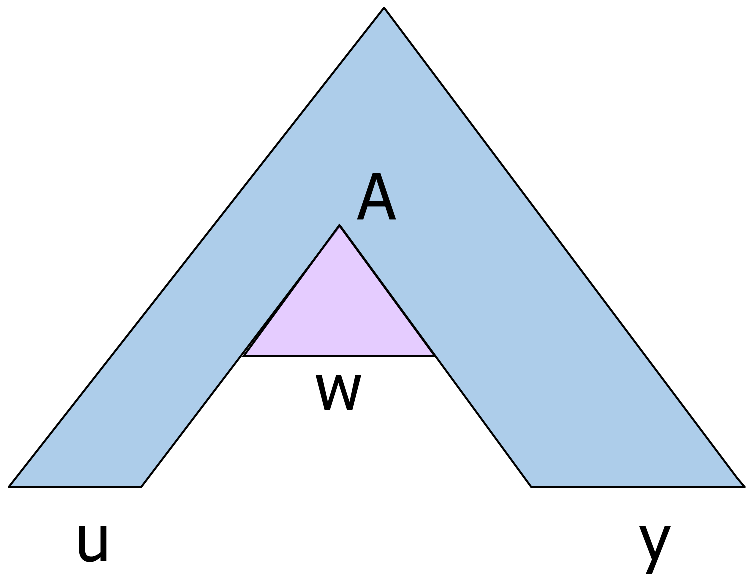
Pump Zero Times

- Case 1: if $A_i=A_j$

This will vanish both v and x

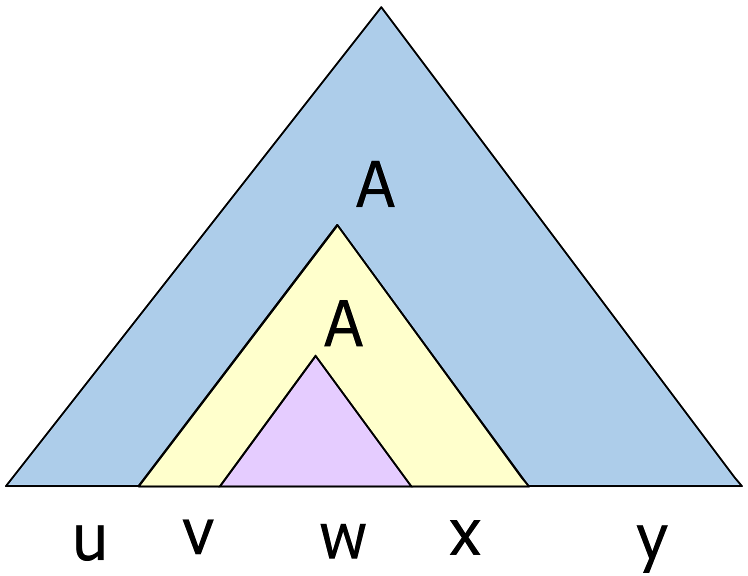
It corresponds to uv^0wx^0y

Hence, string generated is uwy



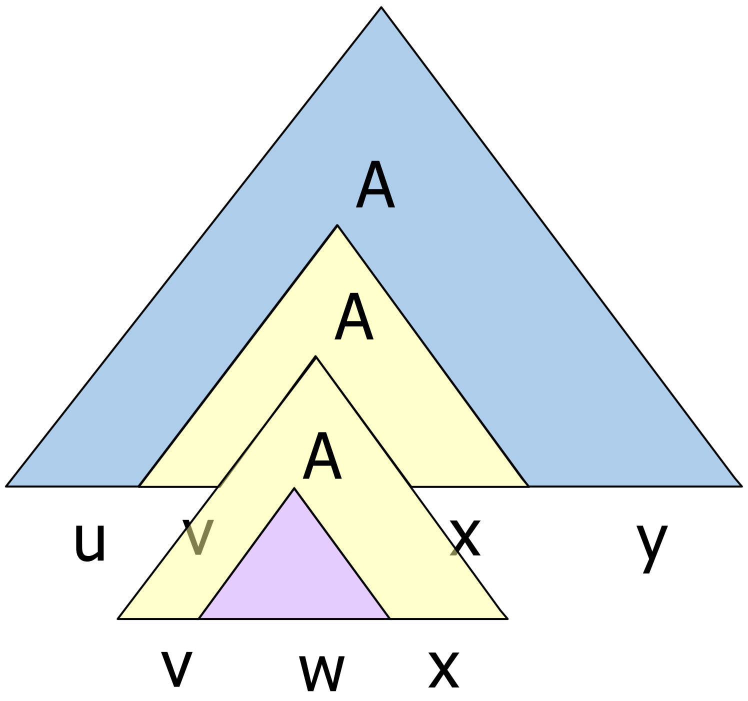
- Case 2:

If v,x is pumped twice, string generated is uv^2wx^2y



- Case 2:

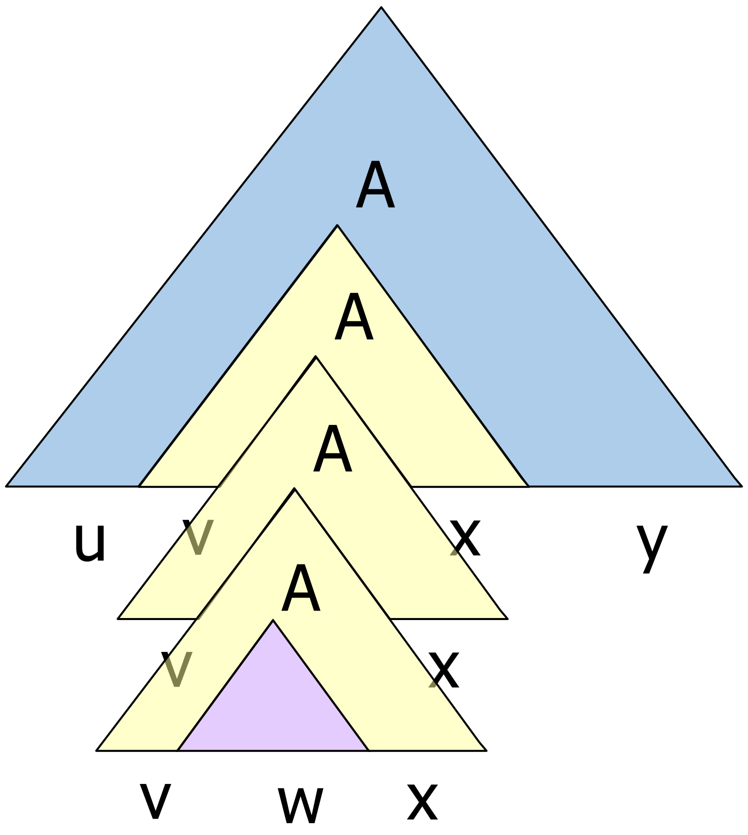
If v,x is pumped twice, string generated is uv^2wx^2y



- Case 2:

If v,x is pumped twice, string generated is uv^2wx^2y

Similarly, v, x, can be pumped any number of times to generate strings uv^iwx^iy, i>1



Example:

- Show that L = {0^n1^n2^n | n>=1} is not context free
- Consider z = $0^n1^n2^n$
- As per pumping lemma we can break it as uvwxy, |vwx|<=n, |vx|>0
- Observation: ‘vwx’ cant have both 0’s and 2’s as they have ‘n’ 1’s in between.
- If ‘vwx’ has no 2’s. Then it should have at least one 0 or 1. Now, as per pumping lemma ‘uwy’ belongs to L. But, in ‘uwy’ there are ‘n’ many 2’s, however less than ‘n’ many 0 or 1.
- Hence, ‘uwy’ cant be in L which is a contradiction. So, L is not CFL.