MA/CSSE 474
Theory of Computation

PDAs and CFGs
Top-down and Bottom-up parsing

But first … another Pumping Theorem example

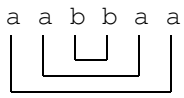
$L = \{a^n b^m a^n,\ n, m \geq 0 \text{ and } n \geq m\}$.

Let $w = a^k b^k a^k$

```
aaa … aaabbb … bbbaaa … aaa
|     1     |     2     |     3     |
```

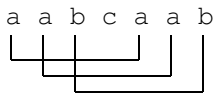
Nested and Cross-Serial Dependencies

PalEven = $\{ww^R : w \in \{a, b\}^*\}$



The dependencies are nested.

WcW = $\{wcw : w \in \{a, b\}^*\}$



Cross-serial dependencies.

Some more examples for you to consider later

- On the next few slides

WcW = {*w*c*w* : *w* ∈ {a, b}*}

Let $w = a^k b^k c a^k b^k$.

```
aaa  ...  aaabbb  ...   bbbcaaa  ...  aaabbb  ...  bbb
|     1     |     2     |3|     4     |     5     |
```

Call the part before c the left side and the part after c the right side.

- If *v* or *y* overlaps region 3, set *q* to 0. The resulting string will no longer contain a c.
- If both *v* and *y* occur before region 3 or they both occur after region 3, then set *q* to 2. One side will be longer than the other.
- If either *v* or *y* overlaps region 1, then set *q* to 2. In order to make the right side match, something would have to be pumped into region 4. Violates $|vxy| \le k$.
- If either *v* or *y* overlaps region 2, then set *q* to 2. In order to make the right side match, something would have to be pumped into region 5. Violates $|vxy| \le k$.

- $\{(ab)^n a^n b^n : n > 0\}$
- $\{x\#y : x, y \in \{0, 1\}^* \text{ and } x \neq y\}$

PDAs and Context-Free Grammars

Theorem: The class of languages accepted by PDAs is exactly the class of context-free languages.

Recall: context-free languages are languages that can be genrated by context-free grammars.

The hard direction: PDA → CFG (later)
The easy direction: CFG → PDA
The idea: Let the stack do the work.

Two approaches:

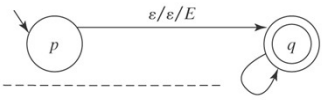
- Top-down
- Bottom-up

Top Down

The idea: Let the stack keep track of expectations.

Example: Arithmetic expressions

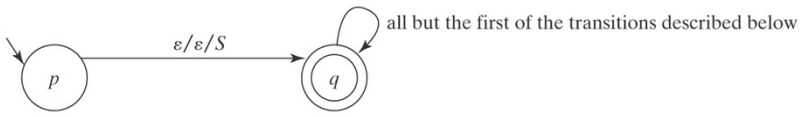
$E \to E + T$
$E \to T$
$T \to T * F$
$T \to F$
$F \to (E)$
$F \to id$



(1) $(q, \varepsilon, E), (q, E+T)$
(2) $(q, \varepsilon, E), (q, T)$
(3) $(q, \varepsilon, T), (q, T*F)$
(4) $(q, \varepsilon, T), (q, F)$
(5) $(q, \varepsilon, F), (q, (E))$
(6) $(q, \varepsilon, F), (q, id)$
(7) $(q, id, id), (q, \varepsilon)$
(8) $(q, (, ( ), (q, \varepsilon)$
(9) $(q,),)), (q, \varepsilon)$
(10) $(q, +, +), (q, \varepsilon)$
(11) $(q, *, *), (q, \varepsilon)$

A Top-Down Parser

The outline of M is:



$M = (\{p, q\}, \Sigma, V, \Delta, p, \{q\})$, where Δ contains:

- The start-up transition $((p, \varepsilon, \varepsilon), (q, S))$.
- For each rule $X \rightarrow s_1 s_2 \ldots s_n$. in R, the transition: $((q, \varepsilon, X), (q, s_1 s_2 \ldots s_n))$.
- For each character $c \in \Sigma$, the transition: $((q, c, c), (q, \varepsilon))$.

Example of the Construction

$L = \{a^n b^* a^n\}$

(1) S → ε *
(2) S → B
(3) S → aSa
(4) B → ε
(5) B → bB

input = a a b b a a

0 (p, ε, ε), (q, S)
1 (q, ε, S), (q, ε)
2 (q, ε, S), (q, B)
3 (q, ε, S), (q, aSa)
4 (q, ε, B), (q, ε)
5 (q, ε, B), (q, bB)
6 (q, a, a), (q, ε)
7 (q, b, b), (q, ε)

This is here for later reference. We did a similar example with the expression grammar.

trans	*state*	*unread input*	*stack*
	p	a a b b a a	ε
0	q	a a b b a a	S
3	q	a a b b a a	aSa
6	q	a b b a a	Sa
3	q	a b b a a	aSaa
6	q	b b a a	Saa
2	q	b b a a	Baa
5	q	b b a a	bBaa
7	q	b a a	Baa
5	q	b a a	bBaa
7	q	a a	Baa
4	q	a a	aa
6	q	a	a
6	q	ε	ε

Another Example : tracing practice

$L = \{a^m b^m c^p d^q : m + n = p + q\}$

(1) $S \rightarrow aSd$
(2) $S \rightarrow T$
(3) $S \rightarrow U$
(4) $T \rightarrow aTc$
(5) $T \rightarrow V$
(6) $U \rightarrow bUd$
(7) $U \rightarrow V$
(8) $V \rightarrow bVc$
(9) $V \rightarrow \varepsilon$

input = a a b c d d

0	$(p, \varepsilon, \varepsilon), (q, S)$
1	$(q, \varepsilon, S), (q, aSd)$
2	$(q, \varepsilon, S), (q, T)$
3	$(q, \varepsilon, S), (q, U)$
4	$(q, \varepsilon, T), (q, aTc)$
5	$(q, \varepsilon, T), (q, V)$
6	$(q, \varepsilon, U), (q, bUd)$
7	$(q, \varepsilon, U), (q, V)$
8	$(q, \varepsilon, V), (q, bVc)$
9	$(q, \varepsilon, V), (q, \varepsilon)$
10	$(q, a, a), (q, \varepsilon)$
11	$(q, b, b), (q, \varepsilon)$
12	$(q, c, c), (q, \varepsilon)$
13	$(q, d, d), (q, \varepsilon)$

trans ***state*** ***unread input*** ***stack***

This is here for later reference. We did a similar example with the expression grammar.

Notice the Nondeterminism

Machines constructed with the algorithm are often nondeterministic, even when they needn't be. This happens even with trivial languages.

Example: $A^nB^n = \{a^n b^n : n \geq 0\}$

A grammar for A^nB^n is:

[1] $S \rightarrow aSb$
[2] $S \rightarrow \varepsilon$

A PDA M for A^nB^n is:

(0) $((p, \varepsilon, \varepsilon), (q, S))$
(1) $((q, \varepsilon, S), (q, aSb))$
(2) $((q, \varepsilon, S), (q, \varepsilon))$
(3) $((q, a, a), (q, \varepsilon))$
(4) $((q, b, b), (q, \varepsilon))$

Transitions 1 and 2 make M nondeterministic.

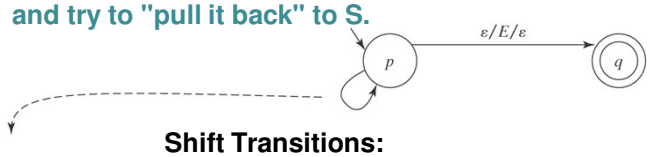
The manually constructed machine for A^nB^n that we created last week is deterministic.

Bottom-Up

The idea: Let the stack keep track of what has been found.

(1) $E \rightarrow E + T$
(2) $E \rightarrow T$
(3) $T \rightarrow T * F$
(4) $T \rightarrow F$
(5) $F \rightarrow (E)$
(6) $F \rightarrow id$

Discover a rightmost derivation in reverse order. Start with the sentence and try to "pull it back" to S.



Reduce Transitions:

(1) $(p, \varepsilon, T + E), (p, E)$
(2) $(p, \varepsilon, T), (p, E)$
(3) $(p, \varepsilon, F * T), (p, T)$
(4) $(p, \varepsilon, F), (p, T)$
(5) $(p, \varepsilon,)E( ), (p, F)$
(6) $(p, \varepsilon, \text{id}), (p, F)$

Shift Transitions:

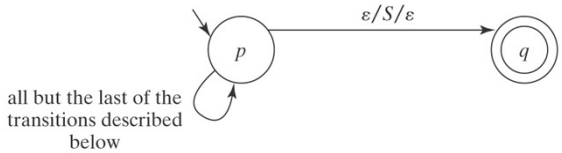
(7) $(p, \text{id}, \varepsilon), (p, \text{id})$
(8) $(p, (, \varepsilon), (p, ()$
(9) $(p,), \varepsilon), (p,))$
(10) $(p, +, \varepsilon), (p, +)$
(11) $(p, *, \varepsilon), (p, *)$

When the right side of a production is on the top of the stack, we can replace it by the left side of that production…

…or not! That's where the nondeterminism comes in: choice between shift and reduce; choice between two reductions.

A Bottom-Up Parser

The outline of M is:



$M = (\{p, q\}, \Sigma, V, \Delta, p, \{q\})$, where Δ contains:

- The shift transitions: $((p, c, \varepsilon), (p, c))$, for each $c \in \Sigma$.
- The reduce transitions: $((p, \varepsilon, (s_1 s_2 \ldots s_n.)^R), (p, X))$, for each rule $X \rightarrow s_1 s_2 \ldots s_n.$ in G.
- The finish up transition: $((p, \varepsilon, S), (q, \varepsilon))$.

Sketch of PDA→CFG

Lemma: If a language is accepted by a pushdown automaton *M*, it is context-free (i.e., it can be described by a context-free grammar).

Proof (by construction):

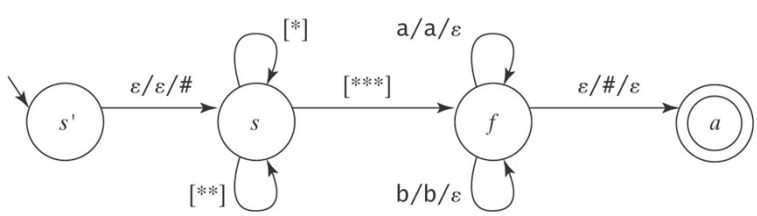
Step 1: Convert *M* to restricted normal form:

- *M* has a start state *s′* that does nothing except push a special symbol # onto the stack and then transfer to a state *s* from which the rest of the computation begins. There must be no transitions back to *s′*.
- *M* has a single accepting state *a*. All transitions into *a* pop # and read no input.
- Every transition in *M*, except the one from *s′*, pops exactly one symbol from the stack.

Second Step - Creating the Productions

Example: WcW^R

M =



The basic idea –

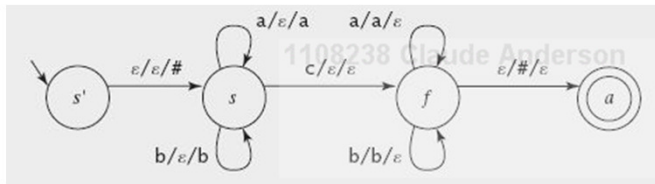
simulate a leftmost derivation of *M* on any input string.

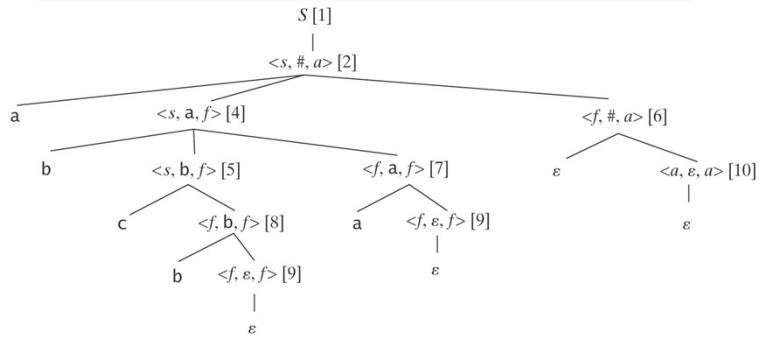
Step 2 - Creating the Productions

The basic idea: A leftmost derivation simulates the actions of M on an input string.

Example:

abcba

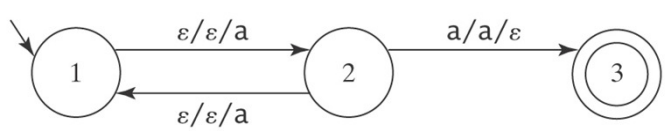




Halting

It is possible that a PDA may
- not halt,
- not ever finish reading its input.

Let Σ = {a} and consider *M* =



L(*M*) = {a}: (1, a, ε) |- (2, a, a) |- (3, ε, ε)

On any other input except a:
- *M* will never halt.
- *M* will never finish reading its input unless its input is ε.

Nondeterminism and Decisions

1. There are context-free languages for which no deterministic PDA exists.

2. It is possible that a PDA may
   - not halt,
   - not ever finish reading its input.
   - require time that is exponential in the length of its input.

3. There is no PDA minimization algorithm.
   It is undecidable whether a PDA is minimal.

Solutions to the Problem

- For NDFSMs:
  - Convert to deterministic, or
  - Simulate all paths in parallel.

- For NDPDAs:
  - No general solution.
  - Formal solutions that usually involve changing the grammar.
    - Such as Chomsky or Greibach Normal form.
  - Practical solutions that:
    - Preserve the structure of the grammar, but
    - Only work on a subset of the CFLs.

Alternative Equivalent Definitions of a PDA

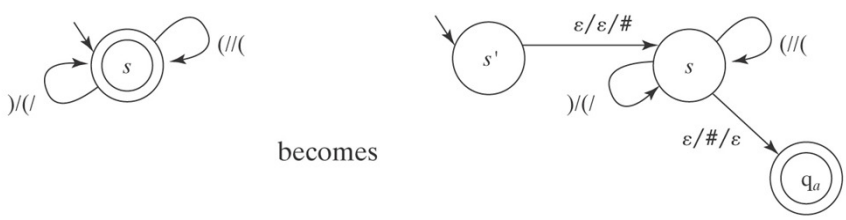
Accept by accepting state at end of string (i.e., we don't care about the stack).

From M (in our definition) we build M' (in this one):

1. Initially, let $M' = M$.
2. Create a new start state s'. Add the transition:
   $((s', \varepsilon, \varepsilon), (s, \#))$.
3. Create a new accepting state q_a.
4. For each accepting state a in M do,
   4.1 Add the transition $((a, \varepsilon, \#), (q_a, \varepsilon))$.
5. Make q_a the only accepting state in M'.

Example

The balanced parentheses language



What About These?

- FSM plus FIFO queue (instead of stack)?
- FSM plus two stacks?

Comparing Regular and Context-Free Languages

Regular Languages	**Context-Free Languages**
• regular exprs. or regular grammars	• context-free grammars
• recognize	• parse
• = DFSMs	• = NDPDAs