MA/CSSE 474
Theory of Computation

Exam Friday; you may bring two sheets of paper.
Material: through HW9.

PDAs and CFGs

Don't forget the faculty candidate talk today:
4:20 O-201

Recap: PDA Definition

$M = (K, \Sigma, \Gamma, \Delta, s, A)$, where:

- K is a finite set of states
- Σ is the input alphabet
- Γ is the stack alphabet
- $s \in K$ is the initial state
- $A \subseteq K$ is the set of accepting states, and
- Δ is the transition relation. It is a finite subset of

Σ and Γ are not necessarily disjoint

$$(K \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^*) \times (K \times \Gamma^*)$$

K	$\Sigma \cup \{\varepsilon\}$	Γ^*	K	Γ^*
state	input or ε	string of symbols to pop from top of stack	state	string of symbols to push on top of stack

Recap: Configurations and Yields

A **configuration** of M is an element of $K \times \Sigma^* \times \Gamma^*$.
An **initial configuration** of M is (s, w, ε), where w is the input string.

Let c be any element of $\Sigma \cup \{\varepsilon\}$,
Let γ_1, γ_2 and γ be any elements of Γ^*, and
Let w be any element of Σ^*.
Then:

$(q_1, cw, \gamma_1\gamma) \mid\!-_M (q_2, w, \gamma_2\gamma)$ iff $((q_1, c, \gamma_1), (q_2, \gamma_2)) \in \Delta$.

Let $\mid\!-_M{}^*$ be the reflexive, transitive closure of $\mid\!-_M$.

C_1 ***yields*** configuration C_2 iff $C_1 \mid\!-_M{}^* C_2$

Yields

Let c be any element of $\Sigma \cup \{\varepsilon\}$,
Let γ_1, γ_2 and γ be any elements of Γ^*, and
Let w be any element of Σ^*.

Then:
$(q_1, cw, \gamma_1\gamma) \mid\!-_M (q_2, w, \gamma_2\gamma)$ iff $((q_1, c, \gamma_1), (q_2, \gamma_2)) \in \Delta$.

Let $\mid\!-_M{}^*$ be the reflexive, transitive closure of $\mid\!-_M$.

C_1 ***yields*** configuration C_2 iff $C_1 \mid\!-_M{}^* C_2$

Recap: Computations and Acceptance

A ***computation*** by M is a finite sequence of configurations C_0, C_1, …, C_n for some $n \geq 0$ such that:

- C_0 is an initial configuration,
- C_n is of the form (q, ε, γ), for some state $q \in K_M$ and some string γ in Γ^*, and
- $C_0 \mid\!-_M C_1 \mid\!-_M C_2 \mid\!-_M \ldots \mid\!-_M C_n$.

A computation C of M is an ***accepting computation*** iff:

$C = (s, w, \varepsilon) \mid\!-_M{}^* (q, \varepsilon, \varepsilon)$, and $q \in A$.

M ***accepts*** a string w iff at least one of its computations accepts.

Other paths may:

- Read all the input and halt in a nonaccepting state,
- Read all the input and halt in an accepting state with the stack not empty,
- Loop forever and never finish reading the input, or
- Reach a dead end where no more input can be read.

The ***language accepted by*** M, denoted $L(M)$, is the set of all strings accepted by M.

Rejecting

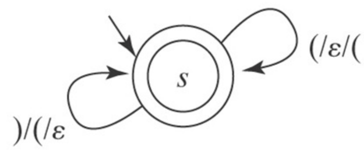
A computation C of M is a ***rejecting computation*** iff:

- $C = (s, w, \varepsilon) \mid\!-_M{}^* (q, w', \alpha)$,
- C is not an accepting computation, and
- M has no moves that it can make from (q, ε, α).

M ***rejects*** a string w iff all of its computations reject.

Note that it is possible that, on input w, M neither accepts nor rejects.

A PDA for Bal

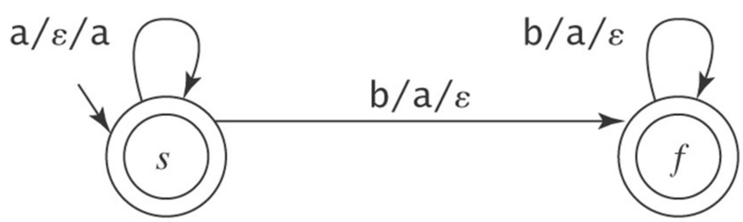


$M = (K, \Sigma, \Gamma, \Delta, s, A)$, where:

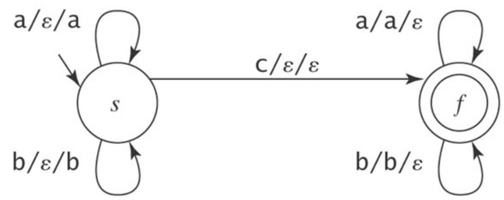
- $K = \{s\}$ the states
- $\Sigma = \{(,)\}$ the input alphabet
- $\Gamma = \{(\}$ the stack alphabet
- $A = \{s\}$
- Δ contains:
  - $((s, (, \varepsilon), (s, ( ))$ **
  - $((s,), ( ), (s, \varepsilon))$

**Important: This does not mean that the stack is empty

A PDA for $A^nB^n = \{a^nb^n: n \geq 0\}$



A PDA for $\{wcw^R: w \in \{a, b\}^*\}$

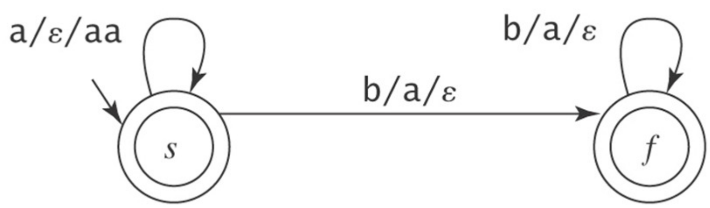


$M = (K, \Sigma, \Gamma, \Delta, s, A)$, where:

$K = \{s, f\}$	the states
$\Sigma = \{a, b, c\}$	the input alphabet
$\Gamma = \{a, b\}$	the stack alphabet
$A = \{f\}$	the accepting states

Δ contains:
$((s, a, \varepsilon), (s, a))$
$((s, b, \varepsilon), (s, b))$
$((s, c, \varepsilon), (f, \varepsilon))$
$((f, a, a), (f, \varepsilon))$
$((f, b, b), (f, \varepsilon))$

A PDA for $\{a^n b^{2n}: n \geq 0\}$

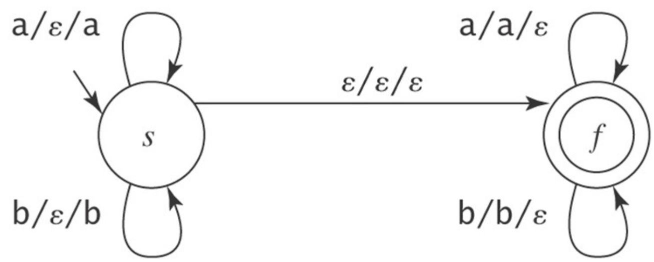


A PDA for PalEven ={ww^R: $w \in$ {a, b}*}

$S \rightarrow \varepsilon$
$S \rightarrow aSa$
$S \rightarrow bSb$

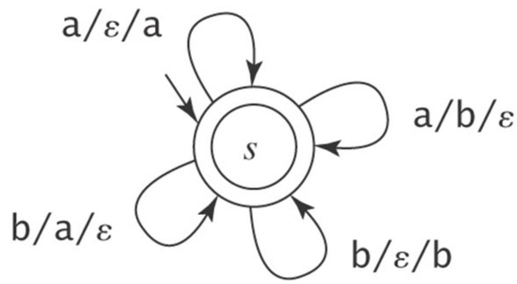
This one is nondeterministic

A PDA:



A PDA for {$w \in$ {a, b}* : $\#_a(w) = \#_b(w)$}

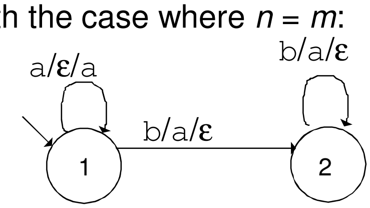
A PDA for $\{w \in \{a, b\}^* : \#_a(w) = \#_b(w)\}$



More on Nondeterminism Accepting Mismatched lengths

$L = \{a^m b^n : m \neq n; m, n > 0\}$

Start with the case where $n = m$:

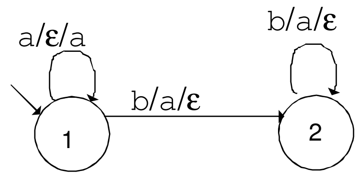


Need to fix it so that

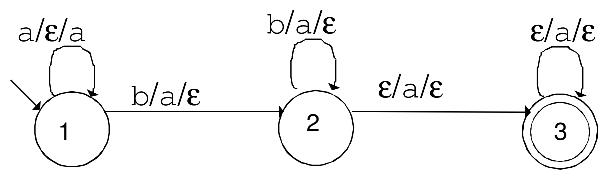
- If stack and input are empty, halt and reject.
- If input is empty but stack is not ($m > n$) (accept):
- If stack is empty but input is not ($m < n$) (accept):

More on Nondeterminism Accepting Mismatches

$L = \{a^m b^n : m \neq n; m, n > 0\}$

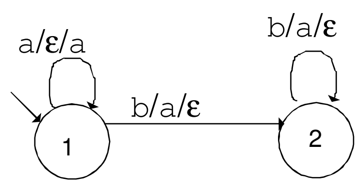


- If input is empty but stack is not ($m < n$) (accept):

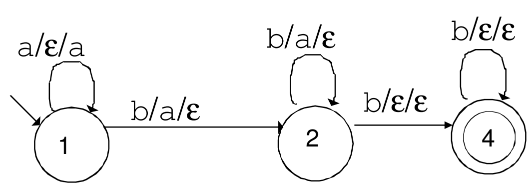


More on Nondeterminism Accepting Mismatches

$L = \{a^m b^n : m \neq n; m, n > 0\}$

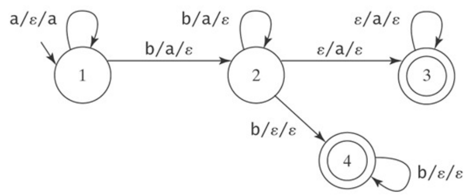


- If stack is empty but input is not ($m > n$) (accept):



Putting It Together

$L = \{a^m b^n : m \neq n;\ m, n > 0\}$



- Jumping to the input-clearing state 4:
  Need to detect bottom of stack.

- Jumping to the stack-clearing state 3:
  Need to detect end of input.

The Power of Nondeterminism

Consider $A^nB^nC^n = \{a^n b^n c^n : n \geq 0\}$.

PDA for it?

The Power of Nondeterminism

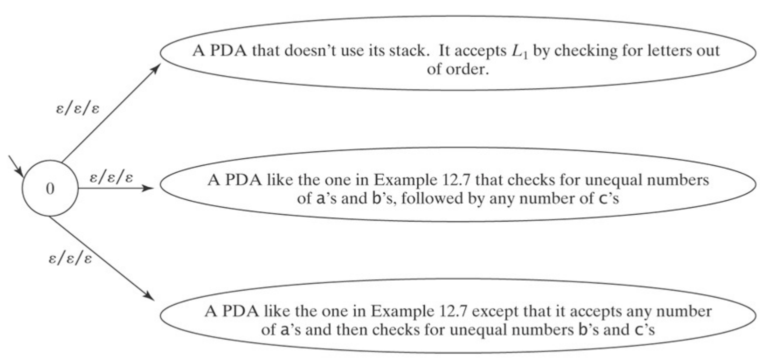
Consider $A^nB^nC^n$ = {$a^nb^nc^n$: $n \geq 0$}.

Now consider $L = \neg A^nB^nC^n$. L is the union of two languages:

1. {$w \in$ {a, b, c}* : the letters are out of order}, and

2. {$a^ib^jc^k$: $i, j, k \geq 0$ and ($i \neq j$ or $j \neq k$)} (in other words, unequal numbers of a's, b's, and c's).

A PDA for $L = \neg A^nB^nC^n$



Are the Context-Free Languages Closed Under Complement?

$\neg A^nB^nC^n$ is context free.

If the CF languages were closed under complement, then

$$\neg\neg A^nB^nC^n = A^nB^nC^n$$

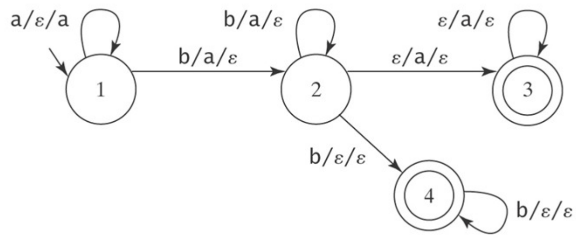
would also be context-free.

But we will prove that it is not.

$L = \{a^nb^mc^p : n, m, p \geq 0 \text{ and } n \neq m \text{ or } m \neq p\}$

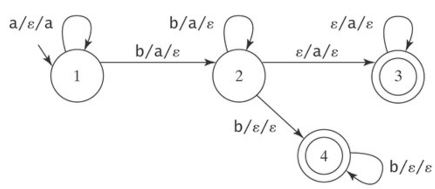
$S \rightarrow NC$	/* $n \neq m$, then arbitrary c's
$S \rightarrow QP$	/* arbitrary a's, then $p \neq m$
$N \rightarrow A$	/* more a's than b's
$N \rightarrow B$	/* more b's than a's
$A \rightarrow a$	
$A \rightarrow aA$	
$A \rightarrow aAb$	
$B \rightarrow b$	
$B \rightarrow Bb$	
$B \rightarrow aBb$	
$C \rightarrow \varepsilon \mid cC$	/* add any number of c's
$P \rightarrow B'$	/* more b's than c's
$P \rightarrow C'$	/* more c's than b's
$B' \rightarrow b$	
$B' \rightarrow bB'$	
$B' \rightarrow bB'c$	
$C' \rightarrow c \mid C'c$	
$C' \rightarrow C'c$	
$C' \rightarrow bC'c$	
$Q \rightarrow \varepsilon \mid aQ$	/* prefix with any number of a's

Reducing Nondeterminism

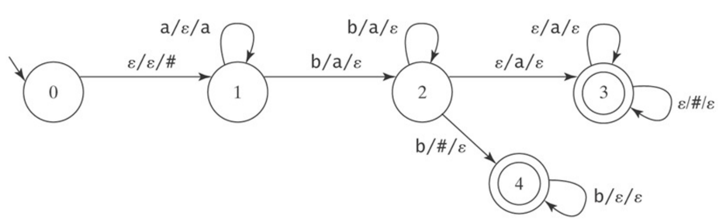


- Jumping to the input-clearing state 4:
  Need to detect bottom of stack, so push # onto the stack before we start.
- Jumping to the stack-clearing state 3:
  Need to detect end of input. Add to *L* a termination character (e.g., $)

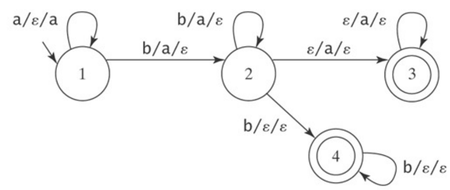
Reducing Nondeterminism



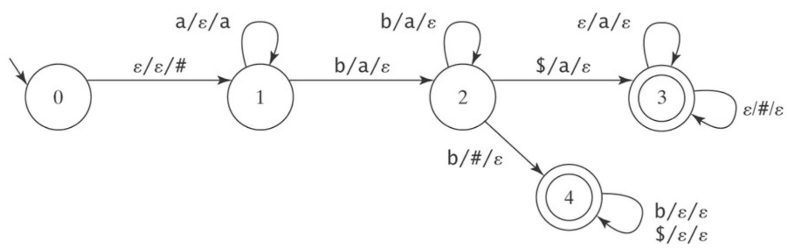
- Jumping to the input-clearing state 4:



Reducing Nondeterminism

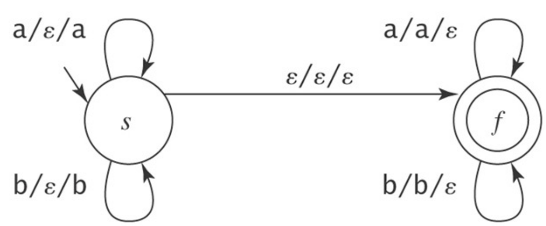


- Jumping to the stack-clearing state 3:



More on PDAs

A PDA for $\{ww^R : w \in \{a, b\}^*\}$:



What about a PDA to accept $\{ww : w \in \{\texttt{a}, \texttt{b}\}^*\}$?

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 defined with context-free grammars.

Restate theorem:

Can describe with context-free grammar

=

Can accept by PDA

Going One Way

Lemma: Each context-free language is accepted by some PDA.

Proof (by construction):

The idea: Let the stack do the work.

Two approaches:

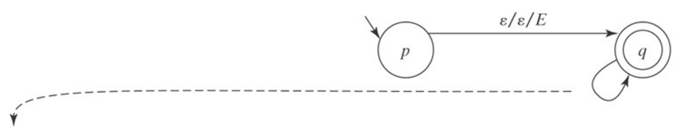
- Top down
- Bottom up

Top Down

The idea: Let the stack keep track of expectations.

Example: Arithmetic expressions

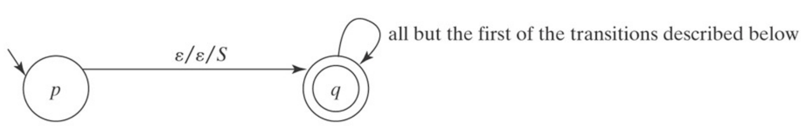
$E \rightarrow E + T$
$E \rightarrow T$
$T \rightarrow T * F$
$T \rightarrow F$
$F \rightarrow (E)$
$F \rightarrow id$



(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

*

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)

input = a a b b a a

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

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

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

input = a a b c d d

Another Example

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

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

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

The Other Way to Build a PDA - Directly

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

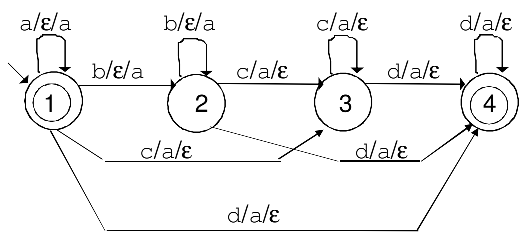
(1) $S \rightarrow aSd$
(2) $S \rightarrow T$
(3) $S \rightarrow U$
(4) $T \rightarrow aTc$
(5) $T \rightarrow V$
(6) $U \rightarrow bUd$
(7) $U \rightarrow V$
(8) $V \rightarrow bVc$
(9) $V \rightarrow \varepsilon$

input = a a b c d d

The Other Way to Build a PDA - Directly

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

(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

Notice 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) $((q, b, b), (q, \varepsilon))$

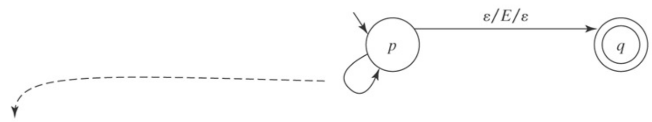
But transitions 1 and 2 make M nondeterministic.

A directly constructed machine for A^nB^n:

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) $F \rightarrow id$

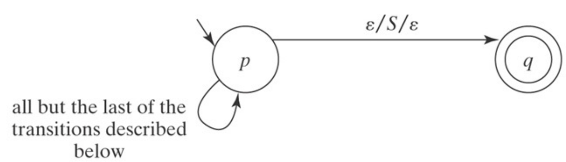


Reduce Transitions:
(1) (*p*, ε, *T* + *E*), (*p*, *E*)
(2) (*p*, ε, *T*), (*p*, *E*)
(3) (*p*, ε, *F* ∗ *T*), (*p*, *T*)
(4) (*p*, ε, *F*), (*p*, *T*)
(5) (*p*, ε,)*E*( ), (*p*, *F*)
(6) (*p*, ε, id), (*p*, *F*)

Shift Transitions
(7) (*p*, id, ε), (*p*, id)
(8) (*p*, (, ε), (*p*, ()
(9) (*p*,), ε), (*p*,))
(10) (*p*, +, ε), (*p*, +)
(11) (*p*, ∗, ε), (*p*, ∗)

A Bottom-Up Parser

The outline of *M* is:



M = ({*p*, *q*}, Σ, *V*, Δ, *p*, {*q*}), where Δ contains:

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