Guilherme Vilatoro Taglianeti
Varadarajan
4/28/2023

First I changed E1 and T1, to G and H respectively. This will remove any ambiguity in the language, and it will make it easier to read.

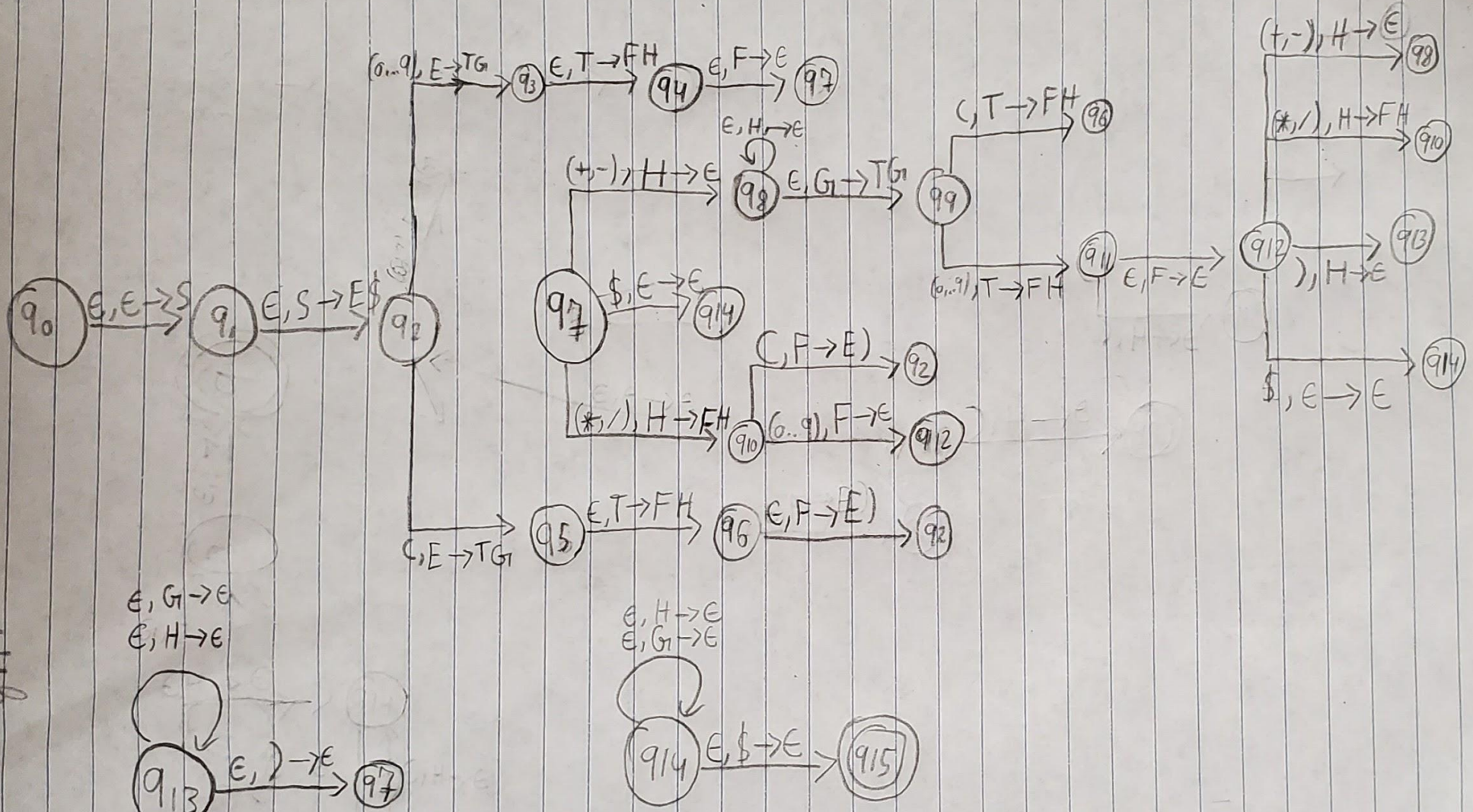
S→E$
E→TG
G→+TG|−TG|∈
T→FH
H→∗FH|/FH|∈
F→(E)| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Now I decided to make a LL(1) parse table so that I can build my DPDA off of it. I also created a symbol called id = 0,1,2,3,4,5,6,7,8,9.

	First	Follow
S→E$	{ (, id,$}	{$,)}
E→TG	{ (, id}	{$,)}
G→+TG|−TG|∈	{+,-, ∈}	{$,)}
T→FH	{ (, id}	{+,-,$,)}
H→∗FH|/FH|∈	{*,/, ∈}	{+,-,$,)}
F→(E)| id	{ (, id}	{*,/,+,-,$,)}

	id	(	)	+	-	*	/	$
S	S→E$	S→E$						
E	E→TG	E→TG						
G			G→∈	G→+TG	G→-TG			G→∈
T	T→FH	T→FH						
H			H→∈	H→∈	H→∈	H→∗FH	H→/FH	H→∈
F	F→ id	F→(E)						

DPDA design:


q0
ε, ε → $
q1
ε, S → E$
q2
(0..9), E → TG
q3
ε, T → FH
q4
ε, F → ε
q7
ε, H → ε
(+,-), H → ε
q8
ε, G → TG
q9
(, T → FH
q6
(0..9), T → FH
q11
ε, F → ε
q12
(+,-), H → ε
q8
(*,/), H → FH
q10
(0..9), F → ε
q12
), H → ε
q13
$, ε → ε
q14
$, ε → ε
q14
ε, $ → ε
q15
C, T → FH
q6
C, E → TG
q5
ε, T → FH
q6
ε, F → E)
q12
C, F → E)
q12
ε, G → ε
ε, H → ε
q13
ε,) → ε
q7
ε, H → ε
ε, G → ε

Description:
q1-q2: This set of instructions set up the start of the DPDA
q2-q4: Reads a number
q2->q5->q6: Sets up the DPDA if it reads a (
q7->q12: checks if there is a special character (+,-,*,/) and takes appropriate actions. If the input expects a “)” q12 will send it to q13 which will set the stack back to what it was before the first “(“ was read (it will get stuck if no “)” is found in the stack). If q12 sees a $ on the input, that means that it is the end of the input and it will be sent to q14 (if “)” or “(” is found in the stack during the loop, the machine will get stuck and will reject). If (+,-,*,/) is in the input q12 will send it back to either q10 or q8 which means that more numbers are expected.

DPDA implementation:

For the PDA implementation, I will make a few changes to the language to make it a little easier to insert into the machine, and it will look like the following:

$S \rightarrow E\$$
$E \rightarrow TG$
$G \rightarrow +TG \mid \in$ *removed -TG because +TG|-TG take the same path in the PDA
$T \rightarrow FH$
$H \rightarrow *FH \mid \in$ *removed /FH because *FH|/FH take the same path in the PDA
$F \rightarrow (E) \mid i$ *changed all numbers into constant i, which will represent all numbers

These changes will only change the number of transitions that must be written for the DPDA, and will make it much easier to insert in the DPDA program when testing. So, if I were going to do a PDA for the complete language, I would have to replace i once for each number and add a transition for - and / whenever + or * appears.

In the simulation that I did, I used the test cases (i+i)*i$, (i+i)*i+i+(i+i)+i$, and (i+i*i$. The first 2 test cases were accepted, and the last one was rejected. The results can be found in the simulation found below.

Testing the DPDA

Enter number of states :
16
Enter input alphabet as a comma-separated list of symbols :
S,E,T,G,F,H,i,$,+,*,(,)
Enter accepting states as a comma-separated list of integers :
15
transitions for state 0:
Need a transition rule for state 0 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
-
Stack symbol to match and pop (enter - for epsilon):
-
State to transition to :
1
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
S
transitions for state 0:
[eps,eps->S]
Need a transition rule for state 0 ? (y or n)
n
transitions for state 1:
Need a transition rule for state 1 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
-
Stack symbol to match and pop (enter - for epsilon):
S
State to transition to :
2
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
E,$
transitions for state 1:
[eps,S->E$]
Need a transition rule for state 1 ? (y or n)
n
transitions for state 2:

Need a transition rule for state 2 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
i
Stack symbol to match and pop (enter - for epsilon):
E
State to transition to :
3
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
T,G
transitions for state 2:
[i,E->TG]
Need a transition rule for state 2 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
(
Stack symbol to match and pop (enter - for epsilon):
E
State to transition to :
5
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
T,G
transitions for state 2:
[i,E->TG]
[(,E->TG]
Need a transition rule for state 2 ? (y or n)
n
transitions for state 3:
Need a transition rule for state 3 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
-
Stack symbol to match and pop (enter - for epsilon):
T
State to transition to :
4
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):

F,H
transitions for state 3:
[eps,T->FH]
Need a transition rule for state 3 ? (y or n)
n
transitions for state 4:
Need a transition rule for state 4 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
-
Stack symbol to match and pop (enter - for epsilon):
F
State to transition to :
7
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
-
transitions for state 4:
[eps,F->eps]
Need a transition rule for state 4 ? (y or n)
n
transitions for state 5:
Need a transition rule for state 5 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
-
Stack symbol to match and pop (enter - for epsilon):
T
State to transition to :
6
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
F,H
transitions for state 5:
[eps,T->FH]
Need a transition rule for state 5 ? (y or n)
n
transitions for state 6:
Need a transition rule for state 6 ? (y or n)
y

Input Symbol to read (enter - for epsilon):
-
Stack symbol to match and pop (enter - for epsilon):
F
State to transition to :
2
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
E,)
transitions for state 6:
[eps,F->E)]
Need a transition rule for state 6 ? (y or n)
n
transitions for state 7:
Need a transition rule for state 7 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
+
Stack symbol to match and pop (enter - for epsilon):
H
State to transition to :
8
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
-
transitions for state 7:
[+,H->eps]
Need a transition rule for state 7 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
*
Stack symbol to match and pop (enter - for epsilon):
H
State to transition to :
10
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
F,H
transitions for state 7:
[+,H->eps]

[*,H->FH]
Need a transition rule for state 7 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
$
Stack symbol to match and pop (enter - for epsilon):
-
State to transition to :
14
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
-
transitions for state 7:
[+,H->eps]
[*,H->FH]
[$,eps->eps]
Need a transition rule for state 7 ? (y or n)
n
transitions for state 8:
Need a transition rule for state 8 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
-
Stack symbol to match and pop (enter - for epsilon):
H
State to transition to :
8
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
-
transitions for state 8:
[eps,H->eps]
Need a transition rule for state 8 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
-
Stack symbol to match and pop (enter - for epsilon):
G
State to transition to :
9

Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
T,G
transitions for state 8:
[eps,H->eps]
[eps,G->TG]
Need a transition rule for state 8 ? (y or n)
n
transitions for state 9:
Need a transition rule for state 9 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
(
Stack symbol to match and pop (enter - for epsilon):
T
State to transition to :
6
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
F,H
transitions for state 9:
[(,T->FH]
Need a transition rule for state 9 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
i
Stack symbol to match and pop (enter - for epsilon):
T
State to transition to :
11
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
F,H
transitions for state 9:
[(,T->FH]
[i,T->FH]
Need a transition rule for state 9 ? (y or n)
n
transitions for state 10:
Need a transition rule for state 10 ? (y or n)

y
Input Symbol to read (enter - for epsilon):
(
Stack symbol to match and pop (enter - for epsilon):
F
State to transition to :
2
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
E,)
transitions for state 10:
[(,F->E)]
Need a transition rule for state 10 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
i
Stack symbol to match and pop (enter - for epsilon):
F
State to transition to :
12
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
-
transitions for state 10:
[(,F->E)]
[i,F->eps]
Need a transition rule for state 10 ? (y or n)
n
transitions for state 11:
Need a transition rule for state 11 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
-
Stack symbol to match and pop (enter - for epsilon):
F
State to transition to :
12
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
-

transitions for state 11:
[eps,F->eps]
Need a transition rule for state 11 ? (y or n)
n
transitions for state 12:
Need a transition rule for state 12 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
$
Stack symbol to match and pop (enter - for epsilon):
-
State to transition to :
14
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
-
transitions for state 12:
[$,eps->eps]
Need a transition rule for state 12 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
)
Stack symbol to match and pop (enter - for epsilon):
H
State to transition to :
13
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
-
transitions for state 12:
[$,eps->eps]
[),H->eps]
Need a transition rule for state 12 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
*
Stack symbol to match and pop (enter - for epsilon):
H
State to transition to :
10

Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
F,H
transitions for state 12:
[$,eps->eps]
[),H->eps]
[*,H->FH]
Need a transition rule for state 12 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
+
Stack symbol to match and pop (enter - for epsilon):
H
State to transition to :
8
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
-
transitions for state 12:
[$,eps->eps]
[),H->eps]
[*,H->FH]
[+,H->eps]
Need a transition rule for state 12 ? (y or n)
n
transitions for state 13:
Need a transition rule for state 13 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
-
Stack symbol to match and pop (enter - for epsilon):
G
State to transition to :
13
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
-
transitions for state 13:
[eps,G->eps]
Need a transition rule for state 13 ? (y or n)

y
Input Symbol to read (enter - for epsilon):
-
Stack symbol to match and pop (enter - for epsilon):
H
State to transition to :
13
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
-
transitions for state 13:
[eps,G->eps]
[eps,H->eps]
Need a transition rule for state 13 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
-
Stack symbol to match and pop (enter - for epsilon):
)
State to transition to :
7
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
-
transitions for state 13:
[eps,G->eps]
[eps,H->eps]
[eps,)->eps]
Need a transition rule for state 13 ? (y or n)
n
transitions for state 14:
Need a transition rule for state 14 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
-
Stack symbol to match and pop (enter - for epsilon):
G
State to transition to :
14

Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
-
transitions for state 14:
[eps,G->eps]
Need a transition rule for state 14 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
-
Stack symbol to match and pop (enter - for epsilon):
H
State to transition to :
14
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
-
transitions for state 14:
[eps,G->eps]
[eps,H->eps]
Need a transition rule for state 14 ? (y or n)
y
Input Symbol to read (enter - for epsilon):
-
Stack symbol to match and pop (enter - for epsilon):
$
State to transition to :
15
Stack symbols to push as comma separated list, first symbol to top of stack (enter - for epsilon):
-
transitions for state 14:
[eps,G->eps]
[eps,H->eps]
[eps,$->eps]
Need a transition rule for state 14 ? (y or n)
n
transitions for state 15:
Need a transition rule for state 15 ? (y or n)
n
Printing all transitions

state 0:
[eps,eps->S]
state 1:
[eps,S->E$]
state 2:
[i,E->TG]
[(,E->TG]
state 3:
[eps,T->FH]
state 4:
[eps,F->eps]
state 5:
[eps,T->FH]
state 6:
[eps,F->E)]
state 7:
[+,H->eps]
[*,H->FH]
[$,eps->eps]
state 8:
[eps,H->eps]
[eps,G->TG]
state 9:
[(,T->FH]
[i,T->FH]
state 10:
[(,F->E)]
[i,F->eps]
state 11:
[eps,F->eps]
state 12:
[$,eps->eps]
[),H->eps]
[*,H->FH]
[+,H->eps]
state 13:
[eps,G->eps]
[eps,H->eps]
[eps,)->eps]
state 14:

[eps,G->eps]
[eps,H->eps]
[eps,$->eps]
state 15:
Enter an input string to be processed by the PDA:
(i+i)*i$
Accept String (i+i)*i$?True
(q0;(i+i)*i$;eps)--[eps,eps->S]-->(q1;(i+i)*i$;S)--[eps,S->E$]-->(q2;(i+i)*i$;E$)--[(,E->TG]-->(q5;i+i)*i$;TG$)--[eps,T->FH]-->(q6;i+i)*i$;FHG$)--[eps,F->E)]-->(q2;i+i)*i$;E)HG$)--[i,E->TG]-->(q3;+i)*i$;TG)HG$)--[eps,T->FH]-->(q4;+i)*i$;FHG)HG$)--[eps,F->eps]-->(q7;+i)*i$;HG)HG$)--[+,H->eps]-->(q8;i)*i$;G)HG$)--[eps,G->TG]-->(q9;i)*i$;TG)HG$)--[i,T->FH]-->(q11;)*i$;FHG)HG$)--[eps,F->eps]-->(q12;)*i$;HG)HG$)--[),H->eps]-->(q13;*i$;G)HG$)--[eps,G->eps]-->(q13;*i$;)HG$)--[eps,)->eps]-->(q7;*i$;HG$)--[*,H->FH]-->(q10;i$;FHG$)--[i,F->eps]-->(q12;$;HG$)--[$,eps->eps]-->(q14;eps;HG$)--[eps,H->eps]-->(q14;eps;G$)--[eps,G->eps]-->(q14;eps;$)--[eps,$->eps]-->(q15;eps;eps)
Accept String (i+i)*i+i+(i+i)+i$?True
(q0;(i+i)*i+i+(i+i)+i$;eps)--[eps,eps->S]-->(q1;(i+i)*i+i+(i+i)+i$;S)--[eps,S->E$]-->(q2;(i+i)*i+i+(i+i)+i$;E$)--[(,E->TG]-->(q5;i+i)*i+i+(i+i)+i$;TG$)--[eps,T->FH]-->(q6;i+i)*i+i+(i+i)+i$;FHG$)--[eps,F->E)]-->(q2;i+i)*i+i+(i+i)+i$;E)HG$)--[i,E->TG]-->(q3;+i)*i+i+(i+i)+i$;TG)HG$)--[eps,T->FH]-->(q4;+i)*i+i+(i+i)+i$;FHG)HG$)--[eps,F->eps]-->(q7;+i)*i+i+(i+i)+i$;HG)HG$)--[+,H->eps]-->(q8;i)*i+i+(i+i)+i$;G)HG$)--[eps,G->TG]-->(q9;i)*i+i+(i+i)+i$;TG)HG$)--[i,T->FH]-->(q11;)*i+i+(i+i)+i$;FHG)HG$)--[eps,F->eps]-->(q12;)*i+i+(i+i)+i$;HG)HG$)--[),H->eps]-->(q13;*i+i+(i+i)+i$;G)HG$)--[eps,G->eps]-->(q13;*i+i+(i+i)+i$;)HG$)--[eps,)->eps]-->(q7;*i+i+(i+i)+i$;HG$)--[*,H->FH]-->(q10;i+i+(i+i)+i$;FHG$)--[i,F->eps]-->(q12;+i+(i+i)+i$;HG$)--[+,H->eps]-->(q8;i+(i+i)+i$;G$)--[eps,G->TG]-->(q9;i+(i+i)+i$;TG$)--[i,T->FH]-->(q11;+(i+i)+i$;FHG$)--[eps,F->eps]-->(q12;+(i+i)+i$;HG$)--[+,H->eps]-->(q8;(i+i)+i$;G$)--[eps,G->TG]-->(q9;(i+i)+i$;TG$)--[(,T->FH]-->(q6;i+i)+i$;FHG$)--[eps,F->E)]-->(q2;i+i)+i$;E)HG$)--[i,E->TG]-->(q3;+i)+i$;TG)HG$)--[eps,T->FH]-->(q4;+i)+i$;FHG)HG$)--[eps,F->eps]-->(q7;+i)+i$;HG)HG$)--[+,H->eps]-->(q8;i)+i$;G)HG$)--[eps,G->TG]-->(q9;i)+i$;TG)HG$)--[i,T->FH]-->(q11;)+i$;FHG)HG$)--[eps,F->eps]-->(q12;)+i$;HG)HG$)--[),H->eps]-->(q13;+i$;G)HG$)--[eps,G->eps]-->(q13;+i$;)HG$)--[eps,)->eps]-->(q7;+i$;HG$)--[+,H->eps]-->(q8;i$;G$)--[eps,G->TG]-->(q9;i$;TG$)--[i,T->FH]-->(q11;$;FHG$)--[eps,F->eps]-->(q12;$;HG$)--[$,eps->eps]-->(q14;eps;HG$)--[eps,H->eps]-->(q14;eps;G$)--[eps,G->eps]-->(q14;eps;$)--[eps,$->eps]-->(q15;eps;eps)
Enter an input string to be processed by the PDA:
(i+i*i$
Accept String (i+i*i$?False
(q0;(i+i*i$;eps)--[eps,eps->S]-->(q1;(i+i*i$;S)--[eps,S->E$]-->(q2;(i+i*i$;E$)--[(,E->TG]-->(q5;i+i*i$;TG$)--[eps,T->FH]-->(q6;i+i*i$;FHG$)--[eps,F->E)]-->(q2;i+i*i$;E)HG$)--[i,E->TG]-->(q3;+i*i$;TG)HG$)--[eps,T->FH]-->(q4;+i*i$;FHG)HG$)--[eps,F->eps]-->(q7;+i*i$;H

G)HG$)--[+,H->eps]-->(q8;i*i$;G)HG$)--[eps,G->TG]-->(q9;i*i$;TG)HG$)--[i,T->FH]-->(q11;*i$;FHG)HG$)--[eps,F->eps]-->(q12;*i$;HG)HG$)--[*,H->FH]-->(q10;i$;FHG)HG$)--[i,F->eps]-->(q12;$;HG)HG$)--[$,eps->eps]-->(q14;eps;HG)HG$)--[eps,H->eps]-->(q14;eps;G)HG$)--[eps,G->eps]-->(q14;eps;)HG$)