Muhammad Maaz Tariq
2019342
CS-424: Compiler Construction
Assignment 2

Sets Of LR(1) items:

Going step by step here and showing all working. Firstly, augmenting the grammar by adding a new start symbol and a new production.

S' → E

Converting the grammar to an LR(0) grammar by adding a new item for each production.

S' → .E
E → .T-E
E → .T
T → .F×T
T → .F
F → .id

Computing the closure of each LR(0) item.

Closure({S' → .E}) = {S' → .E}
Closure({E → .T-E}) = {E → .T-E, T → .F×T, T → .F, F → .id}
Closure({E → .T}) = {E → .T, T → .F×T, T → .F, F → .id}
Closure({T → .F×T}) = {T → .F×T, F → .id}
Closure({T → .F}) = {T → .F, F → .id}
Closure({F → .id}) = {F → .id}

Computing the GOTO sets for each LR(0) item and each grammar symbol.

GOTO({S' → .E}, E) = {S' → E.}
GOTO({E → .T-E}, T) = {E → T-.E, T → .F×T, T → .F, F → .id}
GOTO({E → .T}, '-') = {E → T-.E, T → .F×T, T → .F, F → .id}
GOTO({T → .F×T}, F) = {T → F.×T}
GOTO({E → T.-E, T → .F×T, T → .F, F → .id}, F) = {T → F.×T}
GOTO({T → F.×T}, T) = {T → F×.T, F → .id}
GOTO({T → F×.T, F → .id}, '-') = {E → T-E.}

Computing the sets of LR(1) items by combining the LR(0) items and their GOTO sets.

I0 = Closure({S' → .E})
I1 = GOTO({S' → .E}, E)
= {S' → E.}
I2 = GOTO({E → .T-E, T → .F×T, T → .F, F → .id}, T)
= {E → T-.E, T → .F×T, T → .F, F → .id}
I3 = GOTO({E → T-.E, T → .F×T, T → .F, F → .id}, F)
= {T → F.×T}
I4 = GOTO({T → F.×T, F → .id}, T)
= {T → F×.T, F → .id}
I5 = GOTO({E → T-E., T → .F×T, T → .F, F → .id}, '-')
= {E → T-E.}

Here are the sets of items:

I0:
S' → .E

I1:
S' → E.

I2:
E → T-.E
E → .T
T → .F×T
T → .F
F → .id

I3:
T → F.×T

I4:
T → F×.T
T → .F
F → .id

I5:
E → T-E.
T → .F×T
T → .F
F → .id

LR(1) Parsing Table:

	id	x	-	$	E	T
0	s2				1	
1				acc		
2	s2	s3	s5			4
3	r5		r5	r5		
4	r4	s3	r4	r4		
5	r3	r3	r3	r3		

Python file attached, and code uploaded on GitHub. Added you as collaborator.

https://github.com/maaz29/Compiler-Assignment-2