SYNTAX ANALYSIS

Lecture 5

PREDICTIVE PARSER

CONTENT

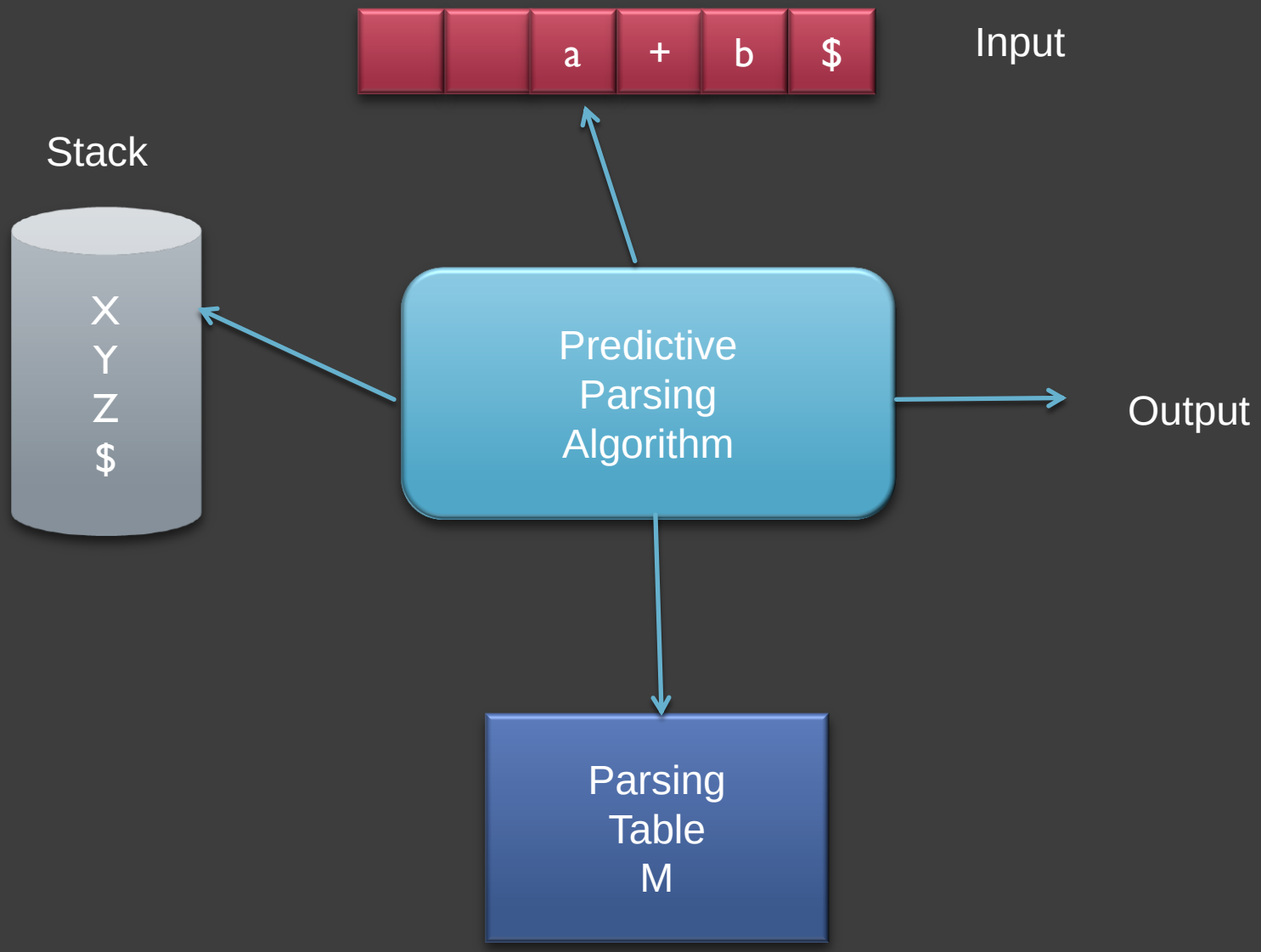
- LL (1) Parser
- Model of Non-Recursive Predictive Parser
- Construction of Predictive Parser Table
- Parsing a string

LL (1) GRAMMAR

LL (1) GRAMMAR

- Used to construct Predictive Parser
- Predictive Parser – Recursive Descent Parser with no need of Backtracking
- First 'L' - scanning input from Left to Right
- Second 'L' - Leftmost Derivation
- "1" - One input symbol of Look ahead at each step to make parsing action decisions
- Left Recursive and Ambiguous grammar is NOT LL(1)

Model Of Non-Recursive Predictive Parser



Construction Of Predictive Parsing Table

Input: Grammar G
Output: Parsing Table M

Table M has Non-Terminals as row and Terminals as columns

For Each Production A → α of grammar do step 1 and 2

Step 1: For each terminal 'a' in FIRST (α)

Add A → α to M [A , a]

Step 2:

Case 1:

If ε is in FIRST (α) then

for each terminal b in FOLLOW (A)

Add A → α to M [A , b]

Case 2:

If ε is in FIRST (α) and $ is in FOLLOW (A) then

for each terminal b in FOLLOW (A)

Add A → α to M [A , b]

Step 3: **Make each undefined entry of M be an error**

Example:

E → TE'

E' → +TE' | ε

T → FT'

T' → *FT' | ε

F → (E) | **id**

	id	+	*	(	)	$
E						
E'						
T						
T'						
F						

Consider Production E → TE'

FIRST (TE') = FIRST (T) = { (, id }

Add E → TE'

to M [E , ( ] and M [E , id]

Example:

E → TE'

E' → +TE' | ε

T → FT'

T' → *FT' | ε

F → (E) | **id**

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'						
T						
T'						
F						

Consider Production E → TE'

FIRST (TE') = FIRST (T) = { (, id }

Add E → TE'

to M [E , ( ] and M [E , id]

Example:

E → TE'

E' → +TE' | ε

T → FT'

T' → *FT' | ε

F → (E) | **id**

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'						
T						
T'						
F						

Consider Production E' → +TE'

FIRST (+TE') = FIRST (+) = { + }

Add E' → +TE' to M [E' , +]

Example:

E → TE'

E' → +TE' | ε

T → FT'

T' → *FT' | ε

F → (E) | **id**

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'				
T						
T'						
F						

Consider Production E' → +TE'

FIRST (+TE') = FIRST (+) = { + }

Add E → +TE' to M [E , +]

Example:

E → TE'

E' → +TE' | ε

T → FT'

T' → *FT' | ε

F → (E) | **id**

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'				
T						
T'						
F						

Consider Production E' → ε

FIRST (ε) = { ε }

Here find FOLLOW (E') = {) , $ }

Add E' → ε to M [E' ,)] and M [E' , $]

Example:

E → TE'

E' → +TE' | ε

T → FT'

T' → *FT' | ε

F → (E) | **id**

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T						
T'						
F						

Consider Production E' → ε

FIRST (ε) = { ε }

Here find FOLLOW (E') = {) , $ }

Add E' → ε to M [E ,)] and M [E , $]

Example:

E → TE'

E' → +TE' | ε

T → FT'

T' → *FT' | ε

F → (E) | **id**

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T						
T'						
F						

Consider Production T → FT'

FIRST (FT') = FIRST (F) = { (, id }

Add T → FT' to M [T , ( ] and M [T , id]

Example:

E → TE'

E' → +TE' | ε

T → FT'

T' → *FT' | ε

F → (E) | **id**

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'						
F						

Consider Production T → FT'

FIRST (FT') = FIRST (F) = { (, id }

Add T → FT' to M [T , ( ] and M [T , id]

Example:

E → TE'

E' → +TE' | ε

T → FT'

T' → *FT' | ε

F → (E) | **id**

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'						
F						

Consider Production T' → *FT'

FIRST (*FT') = FIRST (*) = { * }

Add T' → *FT' to M [T' , *]

Example:

E → TE'

E' → +TE' | ε

T → FT'

T' → *FT' | ε

F → (E) | **id**

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'			T' → *FT'			
F						

Consider Production T' → *FT'

FIRST (*FT') = FIRST (*) = { * }

Add T' → *FT' to M [T' , *]

Example:

E → TE'

E' → +TE' | ε

T → FT'

T' → *FT' | ε

F → (E) | **id**

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'			T' → *FT'			
F						

Consider Production T' → ε

FIRST (ε) = { ε }

Here find FOLLOW (T') = { +,) , $ }

Add E' → ε to M [T', +] , M [T' ,)] and M [T' , $]

Example:

E → TE'

E' → +TE' | ε

T → FT'

T' → *FT' | ε

F → (E) | **id**

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F						

Consider Production T' → ε

FIRST (ε) = { ε }

Here find FOLLOW (T') = { +,) , $ }

Add E' → ε to M [T', +] , M [T' ,)] and M [T' , $]

Example:

E → TE'

E' → +TE' | ε

T → FT'

T' → *FT' | ε

F → (E) | **id**

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F						

Consider Production F → (E)

FIRST ((E)) = FIRST (( ) = { (}

Add F → (E) to M [F, ( ]

Example:

E → TE'

E' → +TE' | ε

T → FT'

T' → *FT' | ε

F → (E) | **id**

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F				F → (E)		

Consider Production F → (E)

FIRST ((E)) = FIRST (( ) = { (}

Add F → (E) to M [F, ( ]

Example:

E → TE'

E' → +TE' | ε

T → FT'

T' → *FT' | ε

F → (E) | **id**

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F				F → (E)		

Consider Production F → id

FIRST (id) = { id }

Add F → id to M [F, id]

Example:

E → TE'

E' → +TE' | ε

T → FT'

T' → *FT' | ε

F → (E) | **id**

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F	F → id			F → (E)		

Consider Production F → id

FIRST (id) = { id }

Add F → id to M [F, id]

Example 2:

S → iEtSS' | a

S' → eS | ε

E → b

	a	b	e	i	t	$
S						
S'						
E						

Consider Production S → iEtSS'

FIRST (iEtSS') = FIRST (i) = { i }

Add S → iEtSS' to M [S , i]

Example 2:

S → iEtSS' | a

S' → eS | ε

E → b

	a	b	e	i	t	$
S				S → iEtSS'		
S'						
E						

Consider Production S → iEtSS'

FIRST (iEtSS') = FIRST (i) = { i }

Add S → iEtSS' to M [S , i]

Example 2:

S → iEtSS' | a

S' → eS | ε

E → b

	a	b	e	i	t	$
S				S → iEtSS'		
S'						
E						

Consider Production S → a

FIRST (a) = { a }

Add S → a to M [S , a]

Example 2:

S → iEtSS' | a

S' → eS | ε

E → b

	a	b	e	i	t	$
S	S → a			S → iEtSS'		
S'						
E						

Consider Production S → a

FIRST (a) = { a }

Add S → a to M [S , a]

Example 2:

S → iEtSS' | a

S' → eS | ε

E → b

	a	b	e	i	t	$
S	S → a			S → iEtSS'		
S'						
E						

Consider Production S' → eS

FIRST (e) = { e }

Add S' → eS to M [S' , e]

Example 2:

S → iEtSS' | a

S' → eS | ε

E → b

	a	b	e	i	t	$
S	S → a			S → iEtSS'		
S'			S' → eS			
E						

Consider Production S' → eS

FIRST (e) = { e }

Add S' → eS to M [S' , e]

Example 2:

S → iEtSS' | a

S' → eS | ε

E → b

	a	b	e	i	t	$
S	S → a			S → iEtSS'		
S'			S' → eS			
E						

Consider Production S' → ε

FIRST (ε) = { ε }

Here find FOLLOW (S') = { e , $ }

Add S' → ε to M [S', e] and M [S' , $]

Example 2:

S → iEtSS' | a

S' → eS | ε

E → b

	a	b	e	i	t	$
S	S → a			S → iEtSS'		
S'			S' → eS S' → ε			S' → ε
E						

Consider Production S' → ε

FIRST (ε) = { ε }

Here find FOLLOW (S') = { e , $ }

Add S' → ε to M [S', e] and M [S' , $]

Example 2:

S → iEtSS' | a

S' → eS | ε

E → b

	a	b	e	i	t	$
S	S → a			S → iEtSS'		
S'			S' → eS S' → ε			S' → ε
E						

Consider Production E → b

FIRST (b) = { b }

Here find FOLLOW (S') = { e , $ }

Add E → b to M [E , b]

Example 2:

S → iEtSS' | a

S' → eS | ε

E → b

	a	b	e	i	t	$
S	S → a			S → iEtSS'		
S'			S' → eS S' → ε			S' → ε
E		E → b				

Consider Production E → b

FIRST (b) = { b }

Here find FOLLOW (S') = { e , $ }

Add E → b to M [E , b]

Construction Of Predictive Parsing Table

- For every LL grammar each parsing table entry uniquely identifies a production or signals an error
- For some grammars however M may have some entries that are multiply defined
- Such grammars are not LL(1) Grammar

Predictive Parsing Algorithm

```
Let a be the first symbol of w
Let X be the top of the Stack symbol
while ( X != $)
{
    if ( X == a)
        pop the stack and let 'a' be the next symbol of w
    else if ( X is a terminal )                    // X != a and X is terminal
        Error ( )
    else if ( M [ X , a ] is an error entry )
        Error ( )
    else if ( M [ X , a ] = Y1 Y2 … Yk )
        Output the production X → Y1 Y2 … Yk
        Pop the stack
        Push Yk Yk-1 … Y1 onto the stack with Y1 on top
    Let X be the top stack symbol
}
```

Predictive Parsing Algorithm

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F	F → id			F → (E)		

Matched	Stack	Input	Action
	E $	id + id * id $	

Predictive Parsing Algorithm

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F	F → id			F → (E)		

Matched	Stack	Input	Action
	E $	**id + id * id $**	
	TE' $	**id + id * id $**	**Output E** → TE'

Predictive Parsing Algorithm

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F	F → id			F → (E)		

Matched	Stack	Input	Action
	E $	**id + id * id $**	
	TE' $	**id + id * id $**	**Output E** → TE'
	FT'E' $	id + id * id $	Output T→ FT'

Predictive Parsing Algorithm

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F	F → id			F → (E)		

Matched	Stack	Input	Action
	E $	**id + id * id $**	
	TE' $	**id + id * id $**	**Output E** → TE'
	FT'E' $	id + id * id $	Output T→ FT'
	idT'E' $	id + id * id $	Output F → id

Predictive Parsing Algorithm

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F	F → id			F → (E)		

Matched	Stack	Input	Action
	E $	**id + id * id $**	
	TE' $	**id + id * id $**	**Output E** → TE'
	FT'E' $	id + id * id $	Output T→ FT'
	idT'E' $	id + id * id $	Output F → id
id	T'E' $	+ id * id $	match id

Predictive Parsing Algorithm

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F	F → id			F → (E)		

Matched	Stack	Input	Action
	E $	**id + id * id $**	
	TE' $	**id + id * id $**	**Output E** → TE'
	FT'E' $	id + id * id $	Output T→ FT'
	idT'E' $	id + id * id $	Output F → id
id	T'E' $	+ id * id $	match id
id	E' $	+ id * id $	Output T' → ε

Predictive Parsing Algorithm

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F	F → id			F → (E)		

Matched	Stack	Input	Action
	E $	**id + id * id $**	
	TE' $	**id + id * id $**	**Output E** → TE'
	FT'E' $	id + id * id $	Output T→ FT'
	idT'E' $	id + id * id $	Output F → id
id	T'E' $	+ id * id $	match id
id	E' $	+ id * id $	Output T' → ε
id	+TE' $	+ id * id $	Output E' → +TE'

Predictive Parsing Algorithm

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F	F → id			F → (E)		

Matched	Stack	Input	Action
id +	TE' $	id * id $	match +

Predictive Parsing Algorithm

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F	F → id			F → (E)		

Matched	Stack	Input	Action
id +	TE' $	id * id $	match +
id +	FT'E' $	id * id $	Output T → FT'

Predictive Parsing Algorithm

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F	F → id			F → (E)		

Matched	Stack	Input	Action
id +	TE' $	id * id $	match +
id +	FT'E' $	id * id $	Output T → FT'
id +	idT'E' $	id * id $	Output F → id

Predictive Parsing Algorithm

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F	F → id			F → (E)		

Matched	Stack	Input	Action
id +	TE' $	id * id $	match +
id +	FT'E' $	id * id $	Output T → FT'
id +	idT'E' $	id * id $	Output F → id
id + id	T'E' $	* id $	match id

Predictive Parsing Algorithm

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F	F → id			F → (E)		

Matched	Stack	Input	Action
id +	TE' $	id * id $	match +
id +	FT'E' $	id * id $	Output T → FT'
id +	idT'E' $	id * id $	Output F → id
id + id	T'E' $	* id $	match id
id + id	*FT'E' $	* id $	Output T'→ *FT'

Predictive Parsing Algorithm

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F	F → id			F → (E)		

Matched	Stack	Input	Action
id +	TE' $	id * id $	match +
id +	FT'E' $	id * id $	Output T → FT'
id +	idT'E' $	id * id $	Output F → id
id + id	T'E' $	* id $	match id
id + id	*FT'E' $	* id $	Output T'→ *FT'
id + id *	FT'E' $	id $	match *

Predictive Parsing Algorithm

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F	F → id			F → (E)		

Matched	Stack	Input	Action
id +	TE' $	id * id $	match +
id +	FT'E' $	id * id $	Output T → FT'
id +	idT'E' $	id * id $	Output F → id
id + id	T'E' $	* id $	match id
id + id	*FT'E' $	* id $	Output T'→ *FT'
id + id *	FT'E' $	id $	match *
id + id *	idT'E' $	id $	Output F → id

Predictive Parsing Algorithm

	id	+	*	(	)	$
E	E → TE'			E → TE'		
E'		E' → +TE'			E' → ε	E' → ε
T	T → FT'			T → FT'		
T'		T' → ε	T' → *FT'		T' → ε	T' → ε
F	F → id			F → (E)		

Matched	Stack	Input	Action
id + id * id	T'E' $	$	match id
id + id * id	E' $	$	Output T' → ε
id + id * id	$	$	Output E' → ε