27 Feb 2025
Thursday

Reduction:

F
|
id * id

reduction process perform হচ্ছে।
id কে F এ reduce হচ্ছে
F → id এই rule টি use হচ্ছে।

- Reduce এর process

⇒ A reduction step replaces a specific substring (matching the body of a production)

⇒ Reduction is opposite of derivation.

- leaf node কে reduce করে production head এ যাওয়া।
- id কে reduce করে F বানানো।
- Used in bottom up parsing.

derivation is used in top down parsing.
- F কে open করে id বানানো।

Handle Pruning:

⇒ A substring that matches the body of the production whose reduction means one step reverse of a rightmost derivation.

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow (E) \mid id$

input string : id * id

- কে reduce হচ্ছে that is our handle.

handle pruning করে এই substring এর জন্য এবং যখন reduction হচ্ছে তখন কোনো handle হচ্ছে তা identify করা।

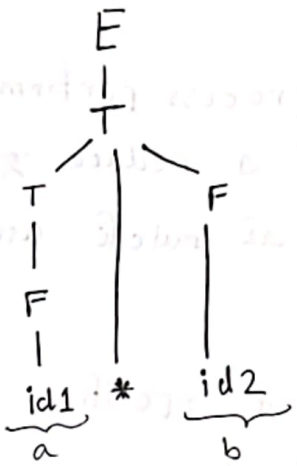
$F \rightarrow id$

id টা F এ reduce হয়েছে

তবে reduce হয়েছে কি

এই phase এ তাদের

handle.



id1 আর id2 এই দুইটার reference আলাদা, Symbol info table এ আলাদা আলাদা object define করা হয়েছে দুইটার জন্য,

Question এ input string id * id দিছিল, এখানে দুইটা id আলাদা আলাদা দুইটা identifier denote করছে, but আমরি initial glance এ " id * id দেখে বলতে পারবো না কোনটা কোন identifier কে denote করছে, so আমরা theoretical concept বুঝার জন্য এই দুইটা id কে আলাদা করে বুঝার জন্য id1, * id2 দিয়ে represent করছি,

Right Sentinental Form	Handle	Production Rule
id 1 * id2	id1	F → id
F * id2	F	T → F
T * id2	id2	F → id
T * F	T * F	T → T * F
T	T	E → T
E		

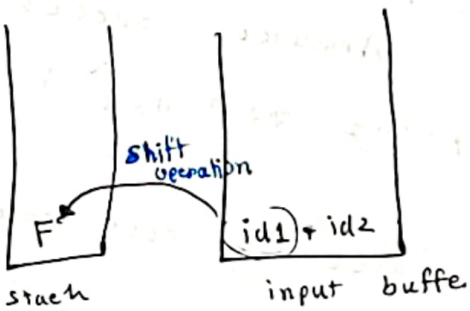
Currently id1 is switched with F as id1 is reduced in F.

Shift Reduce Parsing

- এটা basically parse tree generation এর শেষের stage.
- এই step এ একটা input string gramatically correct কিনা তা check হয়.
- Bottom up parser uses shift reduce parsing.

⇒ Bottom up parsing is also known as shift-reduce parsing; because it's two main actions are shift and reduce.

shift বলতে input buffer থেকে stack এ সাজিয়ে রাখা, buffer এ input string থাকে; সবগুলো শেষ না হয়ে তা stack এ না নিতে পারছি ততক্ষণ পর্যন্ত একাধিকবার operation করতে পারবো না।



Data structure that is being used in this stage is input buffer and stack.

Total 4 operations:

- Shift
- Reduce
- Accept State
- Error State

- যখন সব root node এ পৌঁছায়ে যাচ্ছি তখন accept state
- যখন তার root node এ পৌঁছাতে পারছি না বা root node এ পৌঁছাইছি ঠিকই but input buffer খালি না empty না then error state

Operations :

Shift:

At each shift action, the current symbol in the input string is pushed to the stack

Reduction:

At each reduction step, the symbols at top of the stack will get replaced by the non terminal at the left slide of the production (Production head)

Accept:

Announce successful completion of parsing. (Successfully parsing করতে পারি।)

Successfully root node এ পৌঁছায় গেছি এবং কোনো node parse করা বাকি নাই।

Error:

Discover a syntax error and call error recovery.

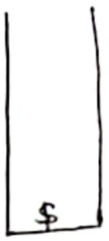
parsing করতে পারি নি বা parsing হইছে ঠিকই but root node এ পৌঁছায়নি so error state.

☒ $E \to E + T \mid T$

$T \to T * F \mid F$

$F \to (E) \mid id$

input string: id1 * id2



$ দিয়ে starting বুঝায়.

input buffer এ $ দিয়ে end of reduce বুঝায়.

Parse tree এ $ এর কাজ আছে.

একটা একটা করে token shift হয়,

⇒

Stack	Input buffer	Action	Handle
$	id1 * id2 $	Shift	
$ id1	* id2 $	Reduce by F → id	id1
$F	* id2 $	Reduce by T → F	F
$T	* id2 $	Shift	
$T *	id2 $	Shift	
$T * id2	$	Reduce by F → id	id2
$T * F	$	Reduce by T → T * F	T * F
$ T	$	Reduce by E → T	T
$E	$	Accept	

এখন buffer empty এবং stack এ root node চলে আসছে successful

as 3 টা আছে so stack top থেকে 3 টা pop করে then push

Successfully root node এ গেছি বলে so accept state

Reduction এর ক্ষেত্রে handle হয়,

Shift এ push হয়

Reduce এ pop হয় then push হয়.

- root node এ চলে আসছি but input buffer still not empty then error state.

- Input buffer is empty but we still couldn't reach the root node so error state