Theory of Automata & Formal Languages

MALAY BHATT

Outline

- Complement of Context-Free Language
- Bottom-Up NPDA Construction

Theorem
Complement of a CFL is not a CFL

- We prove the statement by contradiction
- **Let's assume complement of a CFL is a CFL**
- L1 is a CFL ($L_1 = \{a^i b^j c^k \mid i=j\}$)
- As per assumption, L1' is also a CFL
- L2 is a CFL ($L_2 = \{a^i b^j c^k \mid j=k\}$)
- As per assumption, L2' is also a CFL
- We know that union of two CFLs is also a CFL

  L1' U L2' is also a CFL

There is no need to find L1' or L2'

Theorem

As per assumption, $(L1' \cup L2')'$ is also a CFL

As per DeMorgan's Law,
$(L1' \cup L2')' = (L1')' \cap (L2')' = L1 \cap L2$ which must be CFL.

But, we know that intersection of two CFLs is not always a CFL

$L = L_1 \cap L_2 = \{a^i b^j c^k \mid i=j\} \cap \{a^i b^j c^k \mid j=k\}$
$= \{a^i b^j c^k \mid i=j \text{ and } j=k\}$ is not a CFL

So, out assumption is incorrect. Complement of a CFL is not a CFL

Bottom-Up NPDA

- To construct Bottom-Up NPDA, we will first learn “**Shift-Reduce Parsing**”
- Shift-Reduce Parsing begins with input string

  and step-by-step this technique reduces INPUT STRING and achieves the starting symbol of the Grammar

- It uses Stack Data structure. Initial symbol on the stack is Z_0

Bottom-Up NPDA

Let's consider the Grammar:

S→ S+a

S→ a

Left-most Derivation for the string "a+a" is:

Top- Down	Mirror	Bottom-UP
S		S
S + a		a + S
a + a		a + a

Grammar Rule is in Reverse form on the Stack

S→ S+a
S→ a

Bottom-Up NPDA

Operation	Stack	Unread Input
-	Z0	a + a
Shift	aZ0	+a
Reduce (S→a)	SZ0	+a
Shift	+SZ0	a
Shift	a+SZ0	-
Reduce(S→ S +a)	SZ0	-
	Pop S	
	Pop Z0	

Always Grammar Rule will be in the reverse form on the stack

Bottom-Up NPDA

- Have you observed the following things ?

1. We are shifting every input symbol one by one on the stack
2. If we consider **every possible combination of input symbol with ToS symbol**, then there will be **very large number of shift (Push) rules.**

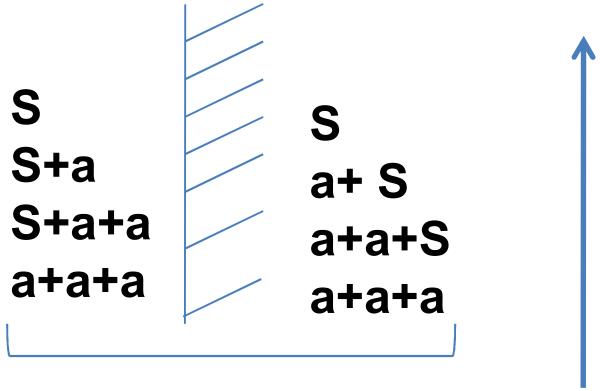
- **To solve this:**

Shift Rule			
STATE	INPUT	ToS	Move
q	σ	X	(q, σX)

S→ S+a
S→ a

Bottom-Up NPDA

Operation	Stack	Unread Input
-	Z0	a+a+a
Shift	aZ0	+a+a
Reduce (S→a)	SZ0	+a+a
Shift	+SZ0	a+a
Shift	a+SZ0	+a
Reduce (S→ S +a)	SZ0	+a
Shift	+SZ0	a
Shift	a+SZ0	-
Reduce (S→ S +a)	SZ0	-
	Pop S	-
	Pop Z0	



Bottom-Up NPDA

- Have you observed the following things ?

1. Which symbol do we **consume** from the **unread input** during the **reduce move** ?
2. We **do not** consume any symbol **or in the other words, we consume ^**
3. **We have to write reduce Rule for every production of the grammar**

- **To solve this: Assign rule number to each grammar production**

Bottom-Up NPDA

1. S→ S+a
2. S→ a

Rule to Reduce a To S			
STATE	INPUT	ToS	Move
q	^	a	(q, S)

Rule to Reduce S+a To S			
STATE	INPUT	ToS	Move
q	^	a	(q1.1, ^)
q1.1	^	+	(q1.2, ^)
q1.2	^	S	(q, S)

S→ S+a
S→ a

Bottom-Up NPDA

Operation	Stack	Unread Input
-	Z0	a+a+a
Shift	aZ0	+a+a
Reduce (S→a)	SZ0	+a+a
Shift	+SZ0	a+a
Shift	a+SZ0	+a
Reduce (S→ S +a)	SZ0	+a
Shift	+SZ0	a
Shift	a+SZ0	-
Reduce (S→ S +a)	SZ0	-
	Pop S	-
	Pop Z0	

S
S+a
S+a+a
a+a+a

S
a+ S
a+a+S
a+a+a

Bottom-Up NPDA

- Have you observed the following things ?

1. Once we are getting the starting symbol (S)of the grammar on the Stack, Which two operations we are performing to print "**Accept**"?
2. **First, we Pop starting symbol of the grammar (S)**
3. **At last, we Pop Z_0**

Rule to Accept			
STATE	INPUT	ToS	Move
q	^	S	(q1, ^)
q1	^	Z_0	(q2, ^)

Pop

Let's combine the whole Example

Shift Rule			
STATE	INPUT	ToS	Move
q	σ	X	(q, σX)

Rule to Reduce a To S			
STATE	INPUT	ToS	Move
q	^	a	(q, S)

1. **S→ S+a**
2. **S→ a**

Rule to Reduce S+a To S			
STATE	INPUT	ToS	Move
q	^	a	(q1.1, ^)
q1.1	^	+	(q1.2, ^)
q1.2	^	S	(q, S)

Rule to Accept			
STATE	INPUT	ToS	Move
q	^	S	(q1, ^)
q1	^	Z_0	(q2, ^)

Bottom-Up NPDA (EXAMPLE-2)

1. **S→ S+T**
2. **S→ T**
3. **T→ T *a**
4. **T→ a**

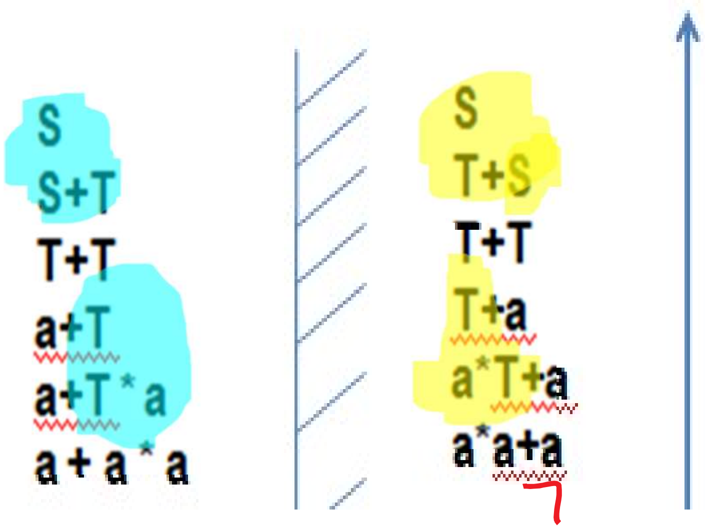
S	**S**
S+T	**T+S**
T+T	**T+T**
a+T	**T+a**
a+T * a	**a*T+a**
a + a * a	**a*a+a**
Top-Down	Bottom-up

Let's derive the String : a + a * a

Bottom-Up NPDA

Operation	Stack	Unread Input
-	Z0	a+a*a
Shift	aZ0	+a*a
Reduce(T→a)	TZ0	+a*a
Reduce(S→T)	SZ0	+a*a
Shift	+SZ0	a*a
Shift	a+SZ0	*a
Reduce(T→a)	T+SZ0	*a
Shift	*T+SZ0	a
Shift	a*T+SZ0	a
Reduce(T→T*a)	T+SZ0	-
Reduce(S→S+T)	SZ0	-
	Pop S	-
	Pop Z0	

1. S→S+T
2. S→T
3. T→T*a
4. T→a



Let’s combine Example-2

Shift Rule			
STATE	INPUT	ToS	Move
q	σ	X	(q, σX)

1. S→ S+T
2. S→ T
3. T→ T *a
4. T→ a

Rule to Accept			
STATE	INPUT	ToS	Move
q	^	S	(q1, ^)
q1	^	Z_0	(q2, ^)

Let's combine Example-2

1. S→ S+T
2. S→ T
3. T→ T *a
4. T→ a

Rule to Reduce T To S			
STATE	INPUT	ToS	Move
q	^	T	(q, S)

Rule to Reduce a To T			
STATE	INPUT	ToS	Move
q	^	a	(q, T)

Rule to Reduce T*a To T			
STATE	INPUT	ToS	Move
q	^	a	(q3.1, ^)
q3.1	^	*	(q3.2, ^)
q3.2	^	T	(q, T)

Rule to Reduce S+T To T			
STATE	INPUT	ToS	Move
q	^	T	(q1.1, ^)
q1.1	^	+	(q1.2, ^)
q1.2	^	S	(q, S)

Bottom-Up NPDA (Example-3)

Shift Rule			
STATE	INPUT	ToS	Move
q	σ	X	(q, σX)

1. S→S[S]
2. S→ ^

Rule to Reduce a To S			
STATE	INPUT	ToS	Move
q	^	^	(q, S)

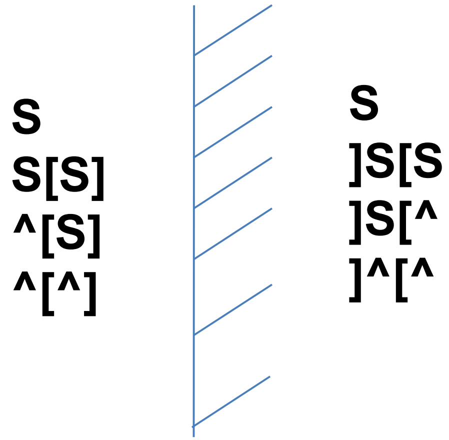
Rule to Reduce S[S] To S			
STATE	INPUT	ToS	Move
q	^	]	(q1.1, ^)
q1.1	^	S	(q1.2, ^)
q1.2	^	[	(q1.3, ^)
q1.3	^	S	(q, S)

Rule to Accept			
STATE	INPUT	ToS	Move
q	^	S	(q1, ^)
q1	^	Z_0	(q2, ^)

Bottom-Up NPDA (Example-3)

Operation	Stack	Unread Input
-	Z0	[]
Reduce(S→^)	SZ0	[]
Shift	[SZ0	]
Reduce(S→^)	S[SZ0	]
Shift	]S[SZ0	-
Reduce(S→S[S])	SZ0	-
	Pop S	-
	Pop Z0	

1. **S→S[S]**
2. **S→ ^**



Bottom-Up NPDA (Example-3)

Operation	Stack	Unread Input
-	Z0	[[]]
Reduce(S→^)	SZ0	[[]]
Shift	[SZ0	[]]
Reduce(S→^)	S[SZ0	[]]
Shift	[S[SZ0	]]
Reduce(S→^)	S[S[SZ0	]]
Shift	]S[S[SZ0	]
Reduce(S→S[S])	S[SZ0	]
Shift	]S[SZ0	-
Reduce(S→S[S])	SZ0	-
	Pop S	-
	Pop Z0	

1. **S→S[S]**
2. **S→ ^**

S	**S**
S[S]	**]S[S**
^[S]	**]S[^**
^[S[S]]	**]]S[s[^**
^[^[S]]	**]]S[^[^**
^[^[^]]	**]]^[^[^**

Practice Problem

- Consider the grammar:
    1. **S→S[S]**
    2. **S→ ^**

Left-most Derivation for the String : [ ] [[ ]] is given below

S

S[S]

S[S][S]

^[S][S]

^[^][S]

^[^][S[S]]

^[^][^[S]]

^[^][^[^]]

Perform Bottom-Up Parsing by generating Shift-Reduce Table