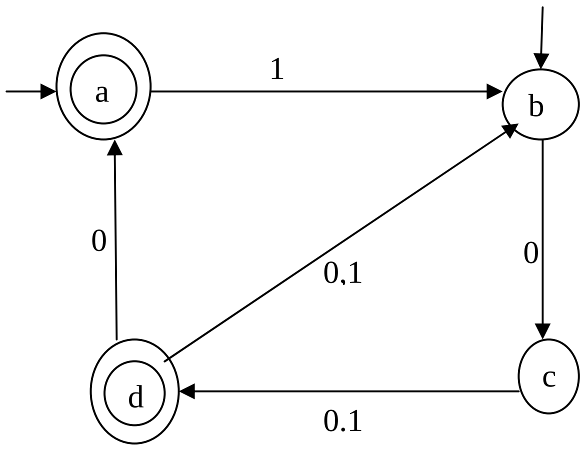
TUTORIAL-8

Exaple-1: Convert the following NFA into a DFA that accepts the same language.

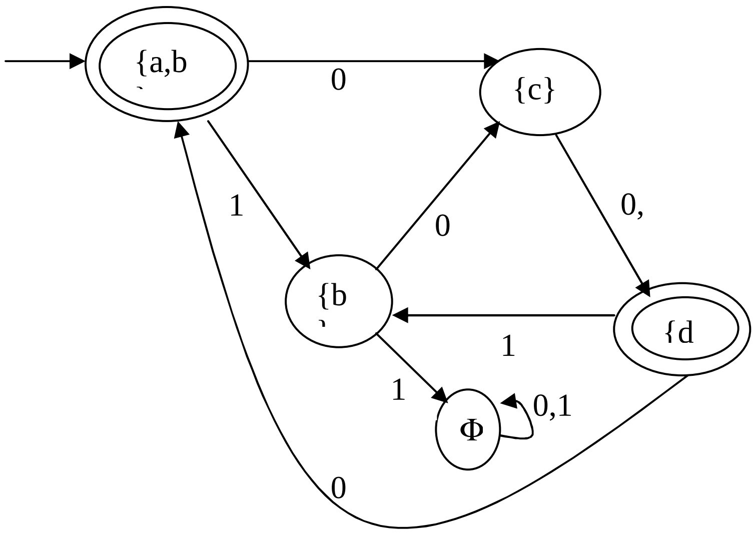


NFA

1. Start state of the DFA is a set of start states for NFA.
2. Create new states for all states reachable from start state.
3. Repeat process for all new states.
4. Stop when no more new states created.

State/Input	0	1
{a, b}	{c}	{b}
{c}	{d}	{d}
{b}	{c}	Φ
{d}	{a, b}	{b}
Φ	Φ	Φ

From the above transition table we have to draw the corresponding DFA.



DFA

Example-2: Consider the following grammar where production (4) denotes a single letter.

```
<E> → <E><E>        (1)
      | <E> : <E>   (2)
      | (<E>)       (3)
      | <letter>    (4)
```

Q: What are the two sources of the ambiguity present in the above grammar?

Ans: (i) Precedence i.e. in this grammar it is not defined that production (1) has got higher precedence or production (2) has got higher precedence or production (3) has got higher precedence.

(ii) Associativity i.e. it is not defined that production (1) or production (2) is left-associative or right-associative (because productions are symmetric).

The above grammar has been re-written into the following unambiguous form.

```
<E> → <T> : <E>     (1)
      | <T>         (2)

<T> → <T> <F>       (3)
      | <F>         (4)

<F> → (<E>)         (5)
      | <letter>    (6)
```

Q: Production (1) or Production (3) has got higher precedence?

Ans: Production (3) has got higher precedence.

Q: State whether Production (1) and Production (3) are right-associative or left-associative?

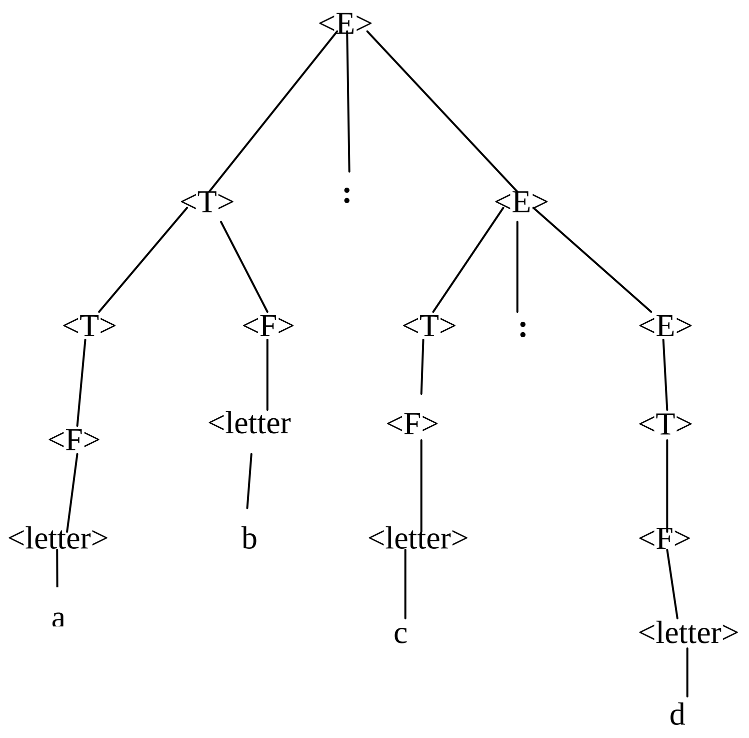
Ans: Production (1) is right-associative and Production (3) is left-associative.

Q: Draw the parse tree for the string "ab:c:d" using the unambiguous grammar.

Ans:

<E> → $\underline{\text{<T>}}$: <E> (1)
→ $\underline{\text{<T>}}$ <F> : <E> (3)
→ $\underline{\text{<F>}}$ <F> : <E> (4)
→ <letter>$\underline{\text{<F> }}$: <E> (6)
→ a<letter> : $\underline{\text{<E>}}$ (6)
→ ab : $\underline{\text{<T>}}$: <E> (1)
→ ab : $\underline{\text{<F>}}$: <E> (4)
→ ab : <letter> : $\underline{\text{<E>}}$ (6)
→ ab : c: $\underline{\text{<T>}}$ (2)
→ ab: c: $\underline{\text{<F>}}$ (4)
→ ab: c : <letter> (6)
→ ab : c : d

PARSE TREE:



Example-3: Consider the following grammar for music expressions where production (1) denotes parallel composition, production (2) denotes sequential composition and production (4) denotes a single note of music.

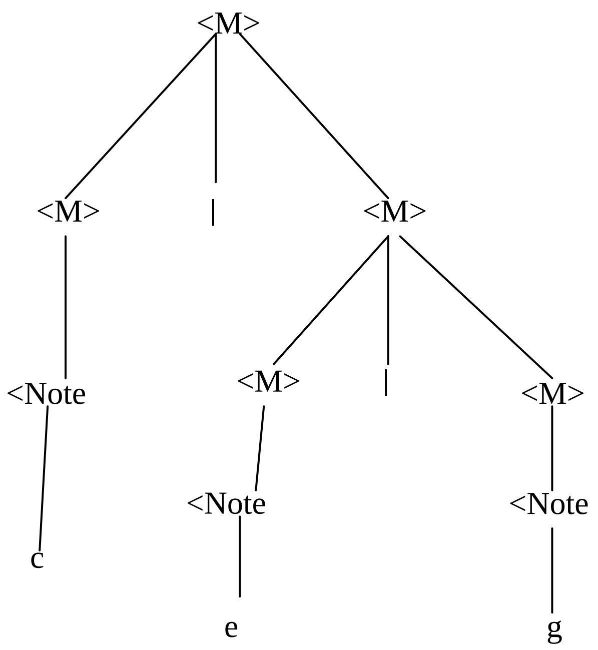
<M> → <M> | <M> (1)
| <M><M> (2)
| (<M>) (3)
| <Note> (4)

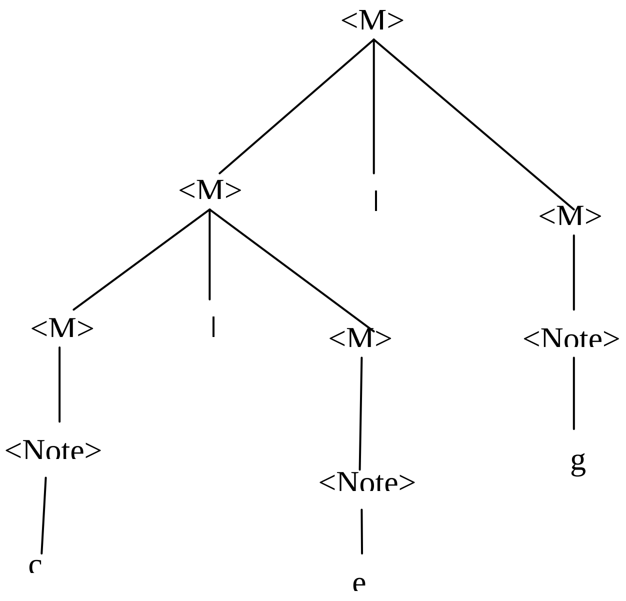
<Note> → a’ | a | a# | b’ | b | c | c# | d’ | d | d# | e’ | e | f | f# | g’ | g | g# | r

Q: Give a string in the language generated by <M> that involves productions (1) to (4) inclusive.

Ans:
<M> → $\underline{\text{<M>}}$ | <M> (1)
→ $\underline{\text{<M>}}$<M> | <M> (2)
→ ($\underline{\text{<M>}}$)<M> | <M> (3)
→ (<Note>)$\underline{\text{<M> }}$| <M> (4)
→ (a’)<Note> | $\underline{\text{<M>}}$ (4)
→ (a’)c# | <Note> (4)
→ (a’)c# | g
String in the language is “(a’)c# | g”

Q: Give two parse trees using the above grammar for the same string “c | e | g”





Q: Re-write the grammar into an unambiguous form, so that parallel composition has a lower precedence to sequential composition and both compositions are left-associative i.e. "c | e | g" is to be interpreted as "(c | e) | g"

Ans:

<M> → <M> | <T>	(1)
| <T>	(2)
<T> → <T><F>	(3)
| <F>	(4)

$$
\begin{aligned}
&\langle F\rangle \rightarrow \langle Note\rangle && (5)\\
&\qquad\quad \mid (\langle M\rangle) && (6)\\
&\langle Note\rangle \rightarrow a' \mid a \mid a\# \mid b' \mid b \mid c \mid c\# \mid d' \mid d \mid d\# \mid e' \mid e \mid f \mid f\# \mid g' \mid g \mid g\# \mid r
\end{aligned}
$$