Prendiamo il seguente esempio:

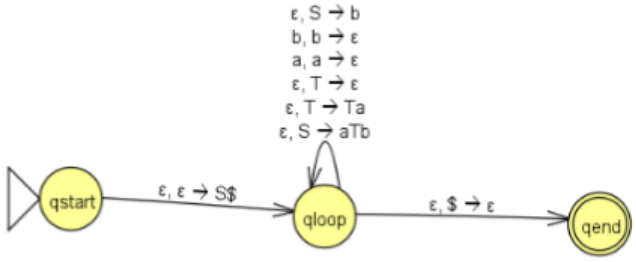
Trasformiamo la seguente CFG in PDA:

$$S \to aTb \mid b$$
$$T \to Ta \mid \varepsilon$$

Dobbiamo metterci 3 stati:

q_{start}, q_{loop}, q_{end}

e successivamente diciamo "se c'è simbolo di input nella pila, poi lo rimuovo":

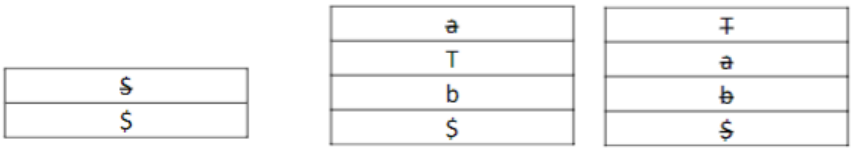


L'idea quindi è di seguire la leftmost derivation, quindi avremmo:

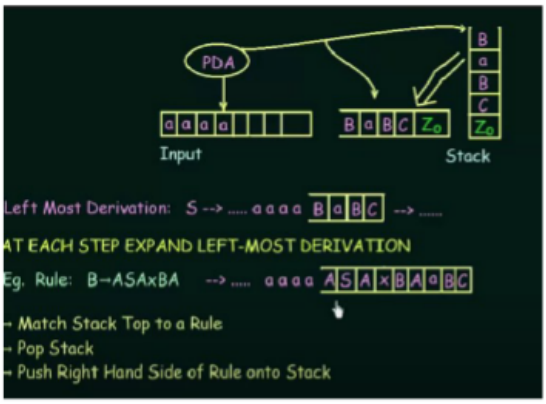
- ε, S → aTb (prima cosa fatta)
- l'altra transizione possibile di S
- le due transizione possibili di T
- a e b (in pop) perché simboli terminali

Essendo grammatica CF, si scelgono liberamente le regole da applicare.

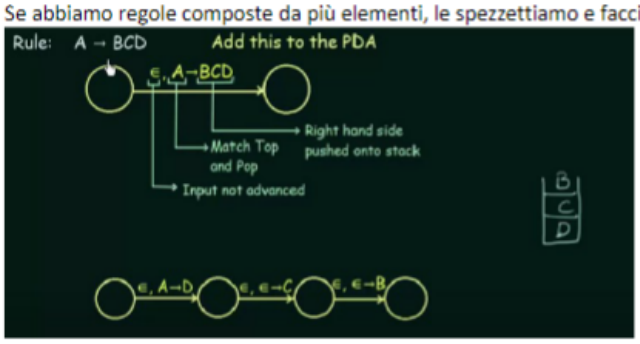
Per esempio scegliamo di attuare una derivazione
S → aTb → aTab → aab



In questo caso, quindi, seguiamo la leftmost derivation, buttiamo dentro (push) tutti i simboli nell'ordine detto dopodiché, sapendo che rimarranno nella pila solo "a" e "b", andremo ad eseguirlo il pop se presenti.
In sintesi abbiamo una roba del tipo:



Quindi vogliamo partire sempre da sinistra con le derivazioni, facendo in modo di avere almeno una regola di quel tipo sulla cima dello stack.
Se abbiamo regole composte da più elementi, le spezzettiamo e facciamo *push* in maniera ordinata:



Successivamente, con questa logica, l'elemento che ho buttato dentro per ultimo sarà il primo ad esseree tolto (pop), facendo avanzare l'input finché non raggiungiamo la fine dell'input: