Per il compilatore, la grammatica deve essere completa, quindi dobbiamo specificare tutti gli elementi. L'identificatore è una stringa che deve necessariamente iniziare con un carattere non numerico. Qui ci sono solo i caratteri maiuscoli, ma è molto facile estendere. È da notare che questo ultimo gruppo di grammatiche (N e I) non sono parte del parsing ma dell'analisi lessicale (scanning) per questo non è tipicamente considerata parte della grammatica principale. Ricordiamo che lo scanning prende il programma e lo divide in parole/lessemi che descrivono entità del programma (parole, identificatori, costanti..) e poi lo si guarda come un singolo elemento. Questo è usato anche per raccogliere le informazioni sui tipi, la fase di scan quindi fa l'operazione di convertire una sequenza di caratteri in una sequenza di lessemi, ovvero di elementi corredati dal "tipo". Questo significa che, se ci mettiamo al livello della grammatica dei comandi, allora possiamo trattare semplicemente identificatori e numeri come insiemi noti N = INTEGERS, I = {id, rate, ciao2,…}. Questo è un esempio di come la potenza delle CFG permetta di modificare la grammatica, ottenendo nuove grammatiche equivalenti ma più adatte magari al processo di elaborazione per il quale vengono definite (compilazione). Quindi le CFG permettono di descrivere i linguaggi ad un profondo livello di dettaglio che non è necessario conoscere per utilizzare il linguaggio. Solitamente l'utente deve conoscere la sintassi astratta del linguaggio. Quindi i punti di vista dell'implementatore del dell'utente sono diversi: è necessaria una grammatica abbastanza grossolana magari senza ambiguità per l'utente che è interessato solo all'abstract sintax tree, dove serve sapere se scrivendo in un modo o in un altro ci sono differenze **semantiche**, ma si necessita una grammatica abbastanza fine per evitare problemi di ambiguità ad esempio nell'implementazione di un parser.