Grammatiche e parsing top-down

Il parsing top down di una sentenza corrisponde a generare una derivazione leftmost della sentenza (o, equivalentemente, un albero di derivazione dal nodo radice alle foglie in depth-first da sinistra verso destra).

Top down parsing:

1. Recursive descent o *parsing a discesa ricorsiva* (*può avere backtracking*)
    a. Implementazione: ogni non terminale è codificata in una funzione
2. Predictive o *parsing predittivo* (*"sbircia" il prossimo simbolo nell'input per decidere quale produzione applicare*)
    a. Implementazione: si crea una tabella le cui entry (Non terminale A da sviluppare, prossimo terminale nell'input) restituiscono la produzione A -> ß da applicare

Non tutte le grammatiche sono adatte per la costruzione di un parser top down:

1. Devono essere ben formate:
    a. ogni non terminale deve avere almeno una produzione che la definisca (*altrimenti non saprei con cosa sostituire un non terminale*)
    Caso negativo : A -> a B C, C-> d (B non ha produzione)
    b. Ogni non terminale deve essere raggiungibile dal non terminal iniziale. Ma questa regola ha non importanza nel parsing top-down.
    Caso negativo: S -> S b | a, B -> c (B non è raggiungibile a partire dal simbolo iniziale S -- la produzione B-> c può essere eliminata)
2. Non devono essere ambigue (*dovrei restituire più di un albero di derivazione per alcune frasi*)
    Caso negativo: A -> a B, A -> C b, C -> a, B -> b. (sentenza "a b" ha due alberi di derivazione)
3. Non devono essere ricorsive sinistre (*potrei avere un loop infinito*)
    Caso negativo: A -> A b, A -> c
4. Devono essere fattorizzate a sinistra (*pure "sbirciando" il prossimo simbolo non saprei quale produzione scegliere*)
    Caso negativo: A -> a B, A -> a C, B -> b, C -> c. (se "A" è il simbolo da espandere ed "a" è il simbolo "sbirciato" nell'input comunque non saprei se usare A -> a B o A -> a C. - necessità di backtrack)

Per un parser recursive descend, posso costruire un parser con eventuale backtracking su grammatiche che rispettano necessariamente tutte le regole 1-3

Per un parser predittivo, posso costruire un parser senza backtracking su grammatiche che rispettano necessariamente tutte le regole 1-4. Ne devono rispettare comunque altre tre non mostrate qui (si veda fondo pagina 223). Tutte queste regole insieme rendono la grammatica LL(k).