Top Down Parsing

II

Recursive Descent Parsing

- In many practical cases a top down parser needs no backtrack
- In order that no backtracking is required we must know
  - given **a** the current input symbol
  - and the nonterminal A to be expanded
    - which one of the alternates of the production $A \rightarrow \alpha_1 / \alpha_2 / \ldots \alpha_n$ is the unique alternate that derives a string beginning with **a**
    - i.e the proper alternate is detectable by looking at only the first symbol it derives
    - e.g stmt→**if** c **then** stmt **else** stmt | **while** c **do** stmt |**begin** stat_list **end**
    - If one of the alternates is $\in$ and no alternates derives a string **a** then accept **a**

Recursive Descent Parsing

- A parser that uses a set of recursive procedures to recognize its input with no backtracking is called ***Recursive-Descent*** parser
- However, to the necessity of a recursive language, a tabular implementation of recursive descent , called predictive parsing can be used

Left Factoring

- Often a grammar is not suitable for recursive-descent parsing even if there is no left recursion
- Example

  stmt→**if** c **then** stmt **else** stmt | **if** c **then** stmt
- A useful method for manipulating grammars into a form suitable for recursive-descent parsing is ***left-factoring***
  - It is the process of factoring out the common prefixes of alternates

Left Factoring

- If $A \rightarrow \alpha\beta \mid \alpha\gamma$ are two A-productions
- and the *input* begins with a *nonempty* string derived from α
- Then left-factored the original productions become
  - $A \rightarrow \alpha_1 A`$
  - $A` \rightarrow \beta \mid \gamma$

Example

S→iCtS|iCtSeS|a

C →b

Can be reduced to

S →iCtSS`

S` →eS| ∈

C →b

Transition Diagram

- can be used as a plan for a recursive-descent parser
- In the case of a parser
  - there is one transition diagram for each nonterminal
  - The labels of edges are tokens or nonterminals
  - A transition on a token(terminal) means we should take that transition if that token is the next input symbol
  - Edges may be nonterminals implying transition diagram for that nonterminal should be called

Transition Diagram

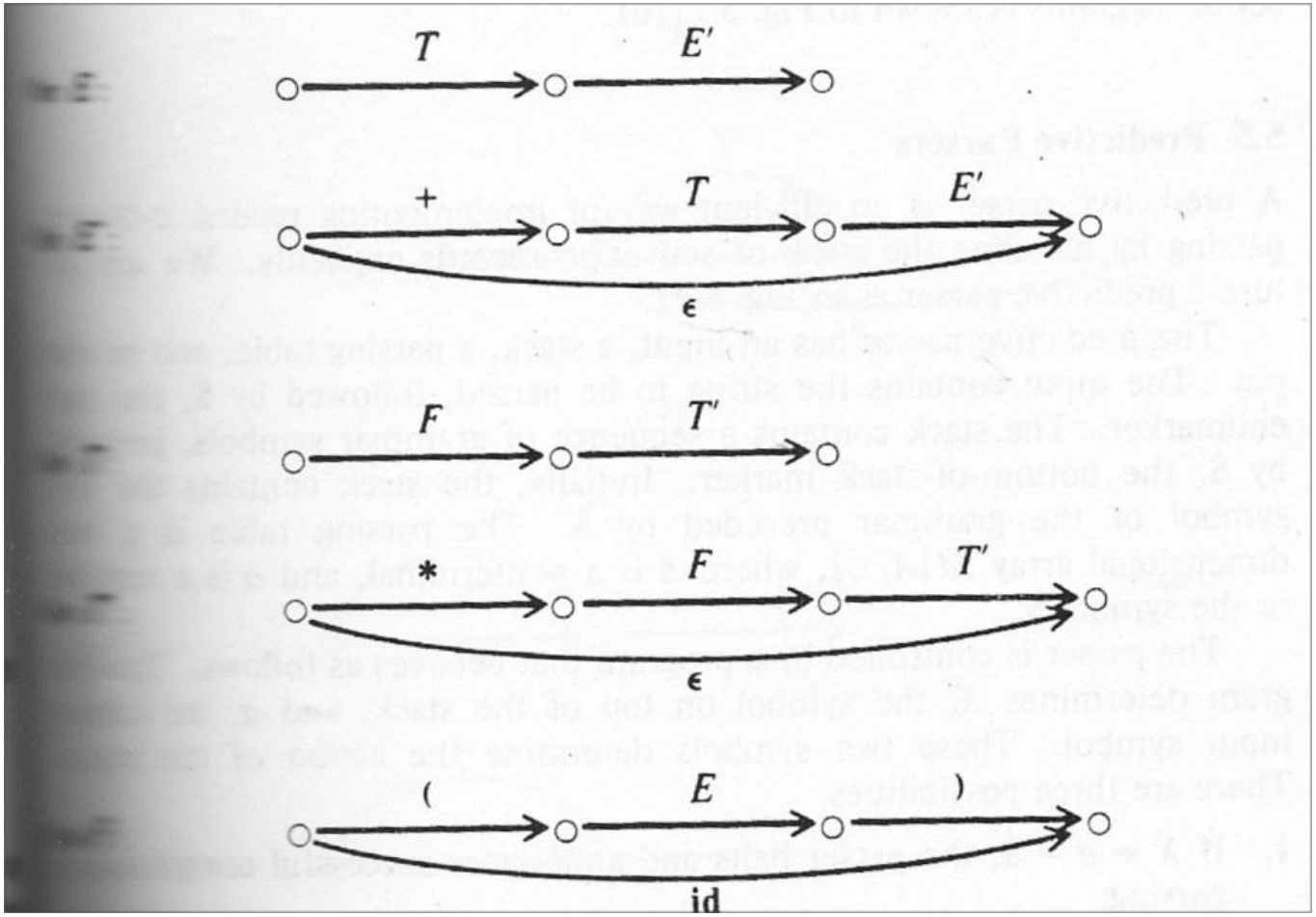
- After removing left recursion and then applying left factoring there's a fair chance of success to choose a path correctly through a transition diagram
- The following can be applied for all nonterminal A

1. Create an initial and final state
2. For each production $A \rightarrow X1,X2,\ldots Xn$ create a path from the initial to the final state with the edges X1,X2,…Xn

Illustration

- $E \rightarrow TE'$
- $E' \rightarrow +TE' \mid \in$
- $T \rightarrow FT'$
- $T' \rightarrow *FT' \mid \in$
- $F \rightarrow (E) \mid id$

Draw Transition Diagrams



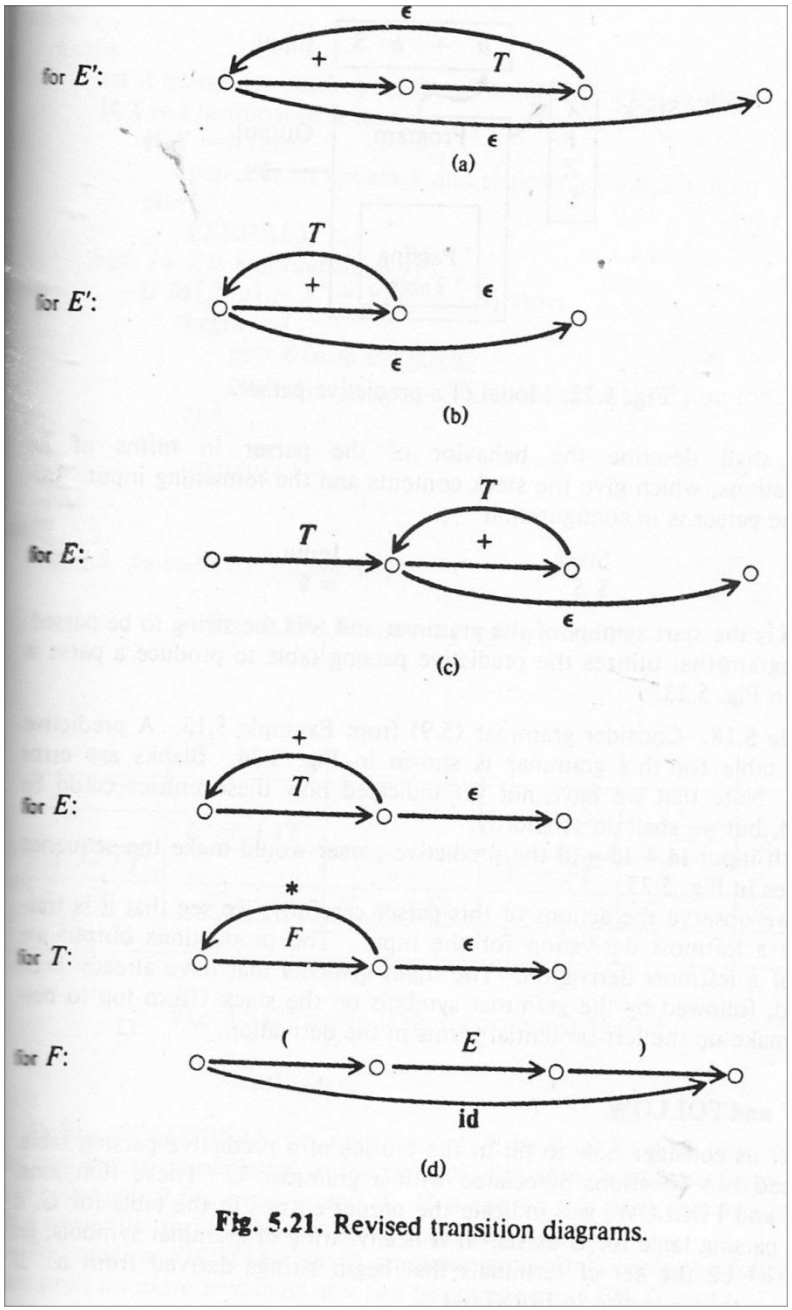


Fig. 5.21. Revised transition diagrams.