Phases of a compiler

**Syntactic structure**

- Character stream
  - Lexical Analyzer
    - Token stream
      - Syntax Analyzer
        - Syntax tree
          - Semantic Analyzer
            - Syntax tree
              - Intermediate Code Generator
                - Intermediate representation
                  - Machine-Independent Code Optimizer
                    - Intermediate representation
                      - Code Generator
                        - Target-machine code
                          - Machine-Dependent Code Optimizer
                            - Target-machine code
Left factoring

- If two (or more) rules share a prefix then their FIRST sets do not distinguish between rule alternatives.
- If there is a choice point later in the rule, rewrite rule by factoring common prefix
- Example: rewrite

\[ A \rightarrow \alpha \beta_1 | \alpha \beta_2 \]

as

\[ A \rightarrow \alpha A' \]

\[ A' \rightarrow \beta_1 | \beta_2 \]
Predictive parsing:
a special case of recursive-descent parsing that does not require backtracking

Each non-terminal $A \in N$ has an associated procedure:

```c
void A() {
    choose an $A$-production $A \rightarrow X_1 X_2 \ldots X_k$
    for (i = 1 to k) {
        if ($x_i \in N$) {
            call $x_i()$
        }
        else if ($x_i = \text{current input symbol}$) {
            advance input to next symbol
        }
        else error
    }
}
```
Predictive parsing:
a special case of recursive-descent parsing that does not require backtracking

Each non-terminal $A \in N$ has an associated procedure:

```c
void A() {
    choose an $A$-production $A \rightarrow X_1 \ X_2 \ ... \ X_k$
    for (i = 1 to k) {
        if ($x_i \in N$) {
            call $x_i()$
        }
        else if ($x_i = \text{current input symbol}$) {
            advance input to next symbol
        }
        else error
    }
}
```

There is non-determinism in choice of production. If "wrong" choice is made the parser will need to revisit its choice by backtracking.

A predictive parser can always make the correct choice here.
if \( X \in T \) then \( \text{FIRST}(X) = \{ X \} \)

if \( X \in N \) and \( X \rightarrow Y_1 Y_2 \ldots Y_k \in P \) for \( k \geq 1 \), then

- add \( a \in T \) to \( \text{FIRST}(X) \) if \( \exists i \) s.t. \( a \in \text{FIRST}(Y_i) \) and \( \varepsilon \in \text{FIRST}(Y_j) \) \( \forall j < i \) (i.e. \( Y_1 Y_2 \ldots Y_{i-1} \Rightarrow^* \varepsilon \))

if \( \varepsilon \in \text{FIRST}(Y_j) \) \( \forall j \leq k \) add \( \varepsilon \) to \( \text{FIRST}(X) \)

if \( X \rightarrow \varepsilon \in P \), then add \( \varepsilon \) to \( \text{FIRST}(X) \)
FOLLOW(X)

- Place $ in FOLLOW(S), where S is the start symbol ($ is an end marker)

- If $A \rightarrow \alpha B\beta \in P$, then FIRST(β) - {ε} is in FOLLOW(B)

- If $A \rightarrow \alpha B \in P$ or $A \rightarrow \alpha B\beta \in P$ where ε ∈ FIRST(β), then everything in FOLLOW(A) is in FOLLOW(B)
Table-driven predictive parsing
Algorithm 4.32 (p. 224)

- INPUT: Grammar $G = (N,T,P,S)$
- OUTPUT: Parsing table $M$
- For each production $A \rightarrow \alpha$ of $G$:
  1. For each terminal $a \in \text{FIRST}(\alpha)$, add $A \rightarrow \alpha$ to $M[A,a]$
  2. If $\epsilon \in \text{FIRST}(\alpha)$, then for each terminal $b$ in $\text{FOLLOW}(A)$, add $A \rightarrow \alpha$ to $M[A,b]$
  3. If $\epsilon \in \text{FIRST}(\alpha)$ and $\$ \in \text{FOLLOW}(A)$, add $A \rightarrow \alpha$ to $M[A,\$]$