Atividade Compiladores

**Implementação de um parser LL1 tabular**

**Aluno: João Pedro do Couto**

**1. Construção dos Conjuntos FIRST**

Para a gramática dada:

* **Produções de S:**
  + S → if E then S else S  
    → O primeiro símbolo é **if**, então **if** ∈ FIRST(S).
  + S → a  
    → O primeiro símbolo é **a**, então **a** ∈ FIRST(S).

**Resultado:**  
**FIRST(S) = {if, a}**

* **Produções de E:**
  + E → b  
    → O primeiro símbolo é **b**.
  + E → c  
    → O primeiro símbolo é **c**.

**Resultado:**  
**FIRST(E) = {b, c}**

**2. Construção dos Conjuntos FOLLOW**

Utilizando as regras:

* Como S é o símbolo inicial, adicionamos o marcador de fim de entrada **$**.
* Para a produção S → if E then S else S:
  + Após o primeiro S vem “else S”. Assim, **else** (o primeiro símbolo do “else S”) vai para FOLLOW(S).
  + Após o segundo S nada vem, mas como S é inicial, já temos o **$**.

**Resultado:**  
**FOLLOW(S) = {else, $}**

* Para E na produção S → if E then S else S:
  + Após E vem “then S else S”. O primeiro símbolo desse trecho é **then**.

**Resultado:**  
**FOLLOW(E) = {then}**

**3. Construção da Tabela LL(1)**

Utilizando os conjuntos FIRST e FOLLOW:

* **Para S:**
  + Produção S → if E then S else S  
    → FIRST dessa produção = {if}  
    → Tabela: M[S, if] = S → if E then S else S.
  + Produção S → a  
    → FIRST dessa produção = {a}  
    → Tabela: M[S, a] = S → a.
* **Para E:**
  + Produção E → b  
    → FIRST = {b}  
    → Tabela: M[E, b] = E → b.
  + Produção E → c  
    → FIRST = {c}  
    → Tabela: M[E, c] = E → c.

A tabela resultante é:

Tela de celular

O conteúdo gerado por IA pode estar incorreto.

**4. Processo de Parsing Manual**

Para a entrada:  
**if b then a else a $**

Utiliza-se uma pilha iniciada com **[$, S]** (sendo o topo da pilha o lado direito) e o marcador **$** adicionado ao final da entrada. Em cada passo:

1. **Pilha:** $ S  
   **Entrada:** if b then a else a $  
   **Ação:** O topo é S, e o lookahead é **if**. Pela tabela, usamos a produção S → if E then S else S.  
   *Ação:* Remover S e empilhar (na ordem inversa) os símbolos: S, else, S, then, E, if.
2. **Pilha:** $ S else S then E if  
   **Entrada:** if b then a else a $  
   **Ação:** O topo é **if** e o lookahead é **if**. Como ambos são terminais, ocorre o *match*: desempilha **if** e avança na entrada.
3. **Pilha:** $ S else S then E  
   **Entrada:** b then a else a $  
   **Ação:** Agora o topo é E (não-terminal) e o lookahead é **b**. Pela tabela, E → b.  
   *Ação:* Desempilha E e empilha **b**.
4. **Pilha:** $ S else S then b  
   **Entrada:** b then a else a $  
   **Ação:** Topo é **b**, que casa com o lookahead **b**. Remove **b** e avança na entrada.
5. **Pilha:** $ S else S then  
   **Entrada:** then a else a $  
   **Ação:** O topo agora é **then**, terminal que casa com o lookahead **then**. Remove **then** e avança.
6. **Pilha:** $ S else S  
   **Entrada:** a else a $  
   **Ação:** Topo é S e lookahead é **a**. Pela tabela, usamos S → a.  
   *Ação:* Desempilha S e empilha **a**.
7. **Pilha:** $ S else a  
   **Entrada:** a else a $  
   **Ação:** Topo é **a**, que casa com o lookahead **a**. Remove **a** e avança.
8. **Pilha:** $ S else  
   **Entrada:** else a $  
   **Ação:** Topo é **else**, que casa com o lookahead **else**. Remove **else** e avança.
9. **Pilha:** $ S  
   **Entrada:** a $  
   **Ação:** Topo é S e lookahead é **a**. Pela tabela, usamos S → a.  
   *Ação:* Desempilha S e empilha **a**.
10. **Pilha:** $ a  
    **Entrada:** a $  
    **Ação:** Topo é **a** e casa com **a** da entrada. Remove **a** e avança.
11. **Pilha:** $  
    **Entrada:** $  
    **Ação:** O topo é **$** e o lookahead também. O parsing termina com sucesso.

**CÓDIGO**

**Obs: O código de exemplo implementado em python, está no outro arquivo( main.py).**

**Funcionamento do código:**

1. **Tabela de Parsing:  
   É definida como um dicionário em que cada chave é um par (não-terminal, terminal) e o valor é a produção correspondente, conforme a tabela construída.**
2. **Pilha e Entrada:  
   A pilha é iniciada com ['$', 'S'] e a entrada é dividida em tokens, com a garantia de que o marcador $ esteja no final.**
3. **Loop Principal:  
   Enquanto a pilha não estiver vazia:**
   * **Se o topo da pilha casa com o lookahead (ambos são terminais), realiza o *match* removendo o símbolo e avançando na entrada.**
   * **Se o topo for um não-terminal, consulta a tabela. Se existir uma produção para o par (não-terminal, lookahead), a produção é aplicada (o não-terminal é removido e os símbolos da produção são empilhados em ordem inversa).**
   * **Se não houver uma produção válida, o parser reporta um erro.**
4. **Finalização:  
   Se ao final a pilha estiver vazia e o lookahead for $, a entrada é aceita.**