



Judson Santos Santiago

Construção do Tradutor

Compiladores


Introdução

- Um tradutor pode ser construído a partir de um esquema de tradução dirigido por sintaxe usando:
 - Análise sintática descendente
percorre a árvore de derivação de cima para baixo
 - Evitando o processo de tentativa e erro
através de um analisador sintático preditivo
 - Trabalhando com uma gramática apropriada
sem recursão à esquerda e usando conjuntos FIRST disjuntos
- Essa técnica é apropriada para um tradutor construído à mão

Introdução

- Um **esquema de tradução dirigido por sintaxe** serve como a especificação de um tradutor

```
expr  expr + term { print('+') }  
    | expr - term { print('-') }  
    | term
```

```
term  0 { print('0') }  
    | 1 { print('1') }  
    | ...  
    | 9 { print('9') }
```


Esquema de tradução
para a cadeias formadas
por **expressões infixadas**


- A **gramática** do esquema de tradução **precisa ser modificada** para ser processada por um analisador sintático preditivo

Recursão à Esquerda

- A técnica de **eliminação da recursão à esquerda** pode ser aplicada em produções contendo **ações semânticas**

```
expr  expr + term { print('+') }  
| expr - term { print('-') }  
| term
```

```
term  0 { print('0') }  
| 1 { print('1') }  
| ...  
| 9 { print('9') }
```

A		A →
		A ↑
		+

A		+R
R		→R
		↑R
		ε

$A = \text{expr}$

$\rightarrow = + \text{ term } \{ \text{print}('+') \}$

$\uparrow = - \text{ term } \{ \text{print}('-') \}$

$+$ = term

Recursão à Esquerda

- **Ações semânticas** são simplesmente copiadas ao longo da transformação, **como se fossem terminais**

```
A  [ ] +R
R  [ ] →R
   | ↑R
   | €
```

$A = \text{expr}$

$\rightarrow = + \text{ term } \{ \text{print}(' + ') \}$

$\uparrow = - \text{ term } \{ \text{print}(' - ') \}$

$+ = \text{term}$

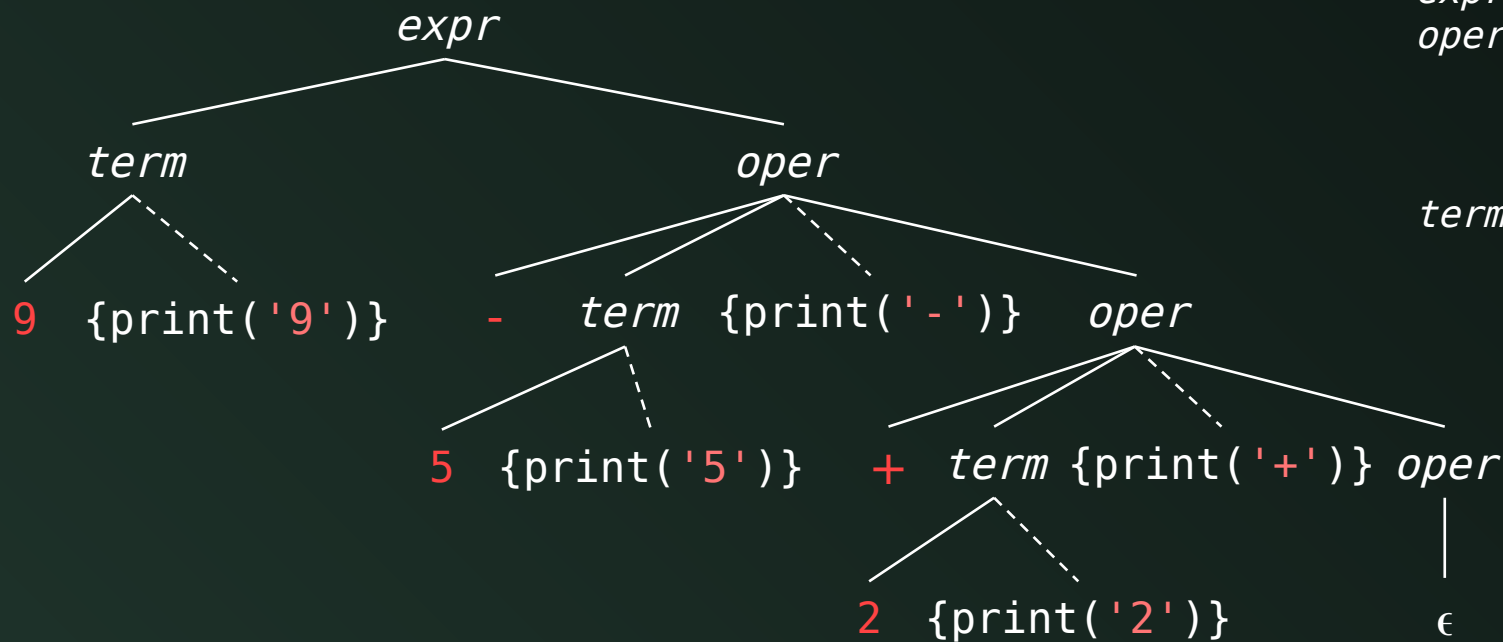
Gramática sem recursão à esquerda

```
expr [ ] term oper
oper [ ] + term { print(' + ') } oper
      | - term { print(' - ') } oper
      | €
```

```
term [ ] 0 { print('0') }
      | 1 { print('1') }
      | ...
      | 9 { print('9') }
```

Recursão à Esquerda

- A árvore de derivação para a entrada **9-5+2**



<i>expr</i>		<i>term oper</i>
<i>oper</i>		<i>+</i> <i>term</i> { <i>print(' + ')</i> } <i>oper</i>
		<i>-</i> <i>term</i> { <i>print(' - ')</i> } <i>oper</i>
		ϵ
<i>term</i>		<i>0</i> { <i>print('0')</i> }
		<i>1</i> { <i>print('1')</i> }
		\dots
		<i>9</i> { <i>print('9')</i> }




Tradutor Dirigido por Sintaxe

```
void expr()
{
    term(); oper();
}

void oper()
{
    if (lookahead == '+')
    { match('+'); term(); print('+'); oper(); }
    else if (lookahead == '-')
    { match('-'); term(); print('-'); oper(); }
    else
    { } // não faz nada com a entrada
}

void term()
{
    if (lookahead é um dígito)
    { t = lookahead; match(lookahead); print(t); }
    else
    { print("syntax error"); }
}
```

Esquema de Tradução

<i>expr</i>		<i>term oper</i>
<i>oper</i>		<i>+</i> <i>term</i> { print('+') }
<i>oper</i>		<i>- term</i> { print('-') }
<i>oper</i>		€
<i>term</i>		<i>0</i> { print('0') }
		<i>1</i> { print('1') }
		...
		<i>9</i> { print('9') }

Tradutor Dirigido por Sintaxe

- Uma **simplificação** pode ser feita em `oper()`

```
void oper()
{
    while(true)
    {
        if (lookahead == '+')
        {
            match('+'); term(); print('+'); continue;
        }
        else if (lookahead == '-')
        {
            match('-'); term(); print('-'); continue;
        }
        break; // não faz nada com a entrada
    }
}
```

Substituindo a
recursão de calda
por um laço while

Tradutor Dirigido por Sintaxe

- Outra **simplificação** pode eliminar `oper()`

```
void expr()
{
    term();

    while(true)
    {
        if (lookahead == '+')
        {
            match('+'); term(); print('+'); continue;
        }
        else if (lookahead == '-')
        {
            match('-'); term(); print('-'); continue;
        }
        break; // não faz nada com a entrada
    }
}
```

Substituindo a
chamada da função
`oper()` em `expr()`
pelo código da função

Resumo

- Um **tradutor** dirigido por sintaxe pode ser obtido:
 1. Escrevendo uma **gramática** para a linguagem fonte
 2. Criando um **esquema de tradução** para a linguagem destino
 3. Adaptando o esquema para um analisador preditivo:
 - Removendo recursão à esquerda
 - Garantindo que os conjuntos FIRST sejam disjuntos
 4. Implementando o **analisador preditivo**
 - Uma função para cada não-terminal
 - Ações semânticas na mesma ordem do esquema de tradução