



Judson Santos Santiago

Tradução Dirigida por Sintaxe

Compiladores

Introdução

- A **sintaxe de uma linguagem** descreve a forma dos programas



Introdução

- As sintaxes das linguagens podem ser representadas por gramáticas
 - Normalmente escritas na notação **Backus-Naur Form (BNF)**

selection_statement

```
: IF '(' expression ')' statement
| IF '(' expression ')' statement ELSE statement
| SWITCH '(' expression ')' statement
;
```

iteration_statement

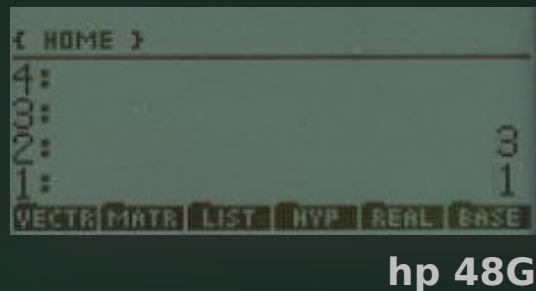
```
: WHILE '(' expression ')' statement
| DO statement WHILE '(' expression ')' ';'
| FOR '(' expression_statement expression_statement expression ')'
statement
;
```

Introdução

- As **gramáticas** podem ser utilizadas para fazer traduções
 - **Tradução dirigida por sintaxe** é uma técnica de compilação
 - Guiada pela gramática da linguagem
 - Ilustra de forma simples a **análise sintática**
- Utilizaremos essa técnica para **introduzir os conceitos básicos**
 - Análise léxica
 - Análise sintática
 - Análise semântica
 - Geração de código intermediário

Tradutor

- A construção de um **tradutor simples dirigido por sintaxe** ajudará a compreender melhor o *front-end* de um compilador
- Ele deve traduzir expressões aritméticas **infixadas** para a forma **pós-fixada**
 - **Notação infixada**: operadores aparecem entre os operandos
 - **Notação pós-fixada**: operadores aparecem após operandos



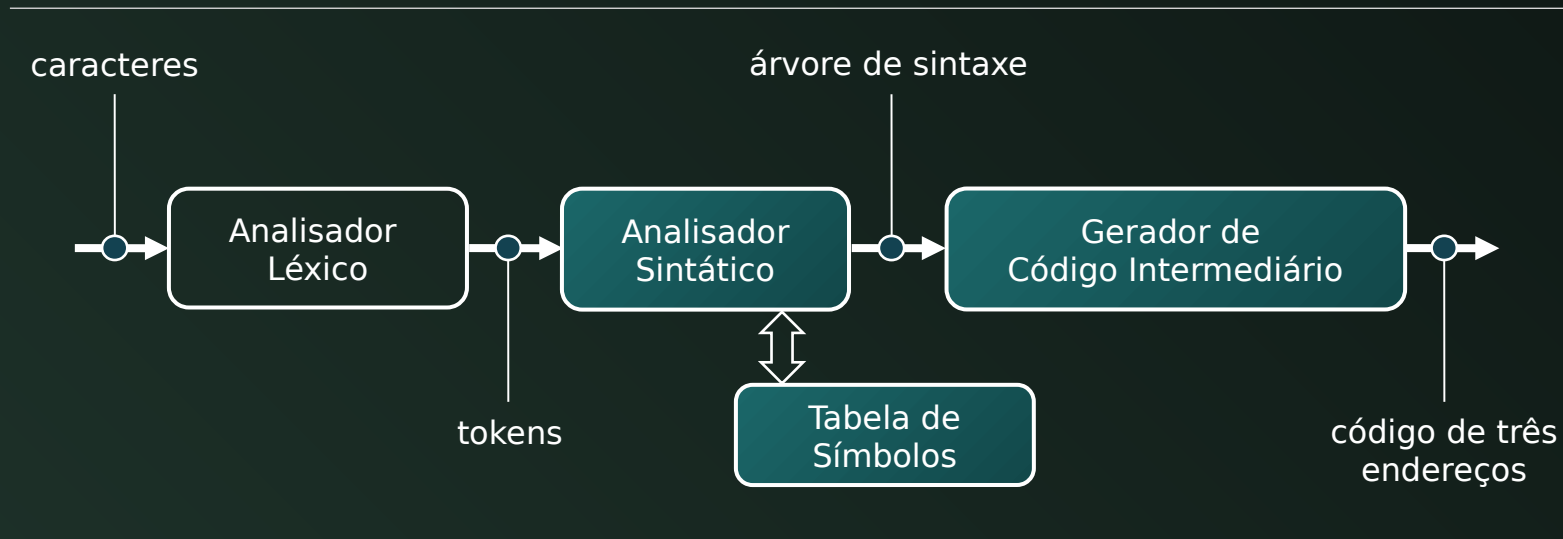
$9 - 5 + 2$ (infixada)



$9\ 5\ -\ 2\ +$ (pós-fixada)

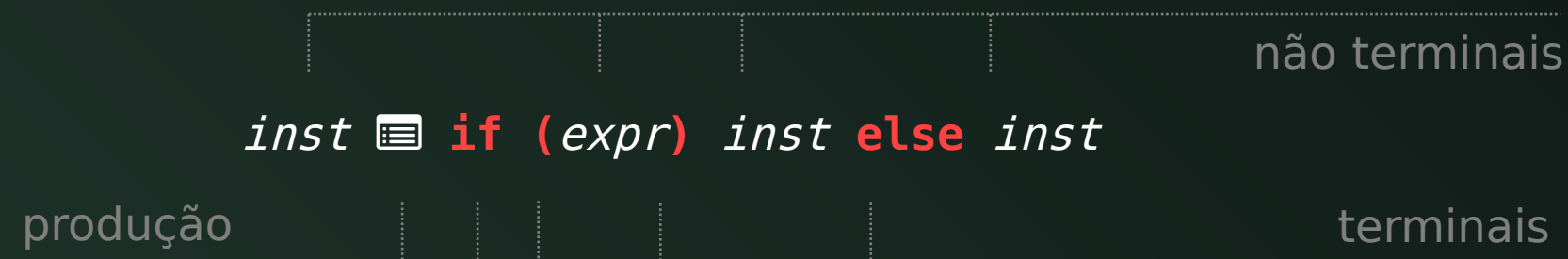
Tradutor

- As expressões serão compostas por dígitos, somas e subtrações
 - Os dígitos serão caracteres individuais (ex.: ✓1 , ✓5 , ✗42)
 - Um analisador léxico não será necessário





Definição da Sintaxe

- Uma **gramática descreve** a estrutura das **construções** das linguagens de programação
 - A construção: **if** (expressão) instrução **else** instrução
 - Pode ser representada em uma gramática pela **produção**:



Definição da Sintaxe

- Uma **gramática livre de contexto** possui quatro **componentes**:
 - Um conjunto de símbolos terminais (**tokens**)
 - Um conjunto de símbolos não-terminais (**variáveis sintáticas**)
 - Um conjunto de **produções**
 - Uma cabeça: símbolo não-terminal
 - Uma seta ()
 - Um corpo: uma sequência de terminais e/ou não-terminais
 - Um não-terminal representando o **símbolo inicial da gramática**

```
inst  if (expr) inst else inst
```

```
.....
```

```
cabeça ..... corpo
```


Definição da Sintaxe

- A **gramática** a seguir **descreve a sintaxe** da nossa linguagem (formada por expressões aritméticas em notação infixada)
 - As expressões são compostas por dígitos, somas e subtrações
 - Os símbolos **+** **-** **0** **1** **2** **3** **4** **5** **6** **7** **8** **9** são **terminais**
 - Os símbolos *expr* e *digit* são **não-terminais**
 - O **símbolo inicial** é sempre a cabeça da primeira produção

```
expr  ::= expr + digit
expr  ::= expr - digit
expr  ::= digit
digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Definição da Sintaxe

- Produções com a mesma cabeça **podem ser agrupadas**

```
expr ::= expr + digit
expr ::= expr - digit
expr ::= digit
digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

O uso da barra vertical (ou) **simplifica a escrita** de

```
expr ::= expr + digit | expr - digit | digit
```

```
expr ::= expr +
      | expr -
      | digit
digit ::= 0 | ... | 9
```

O símbolo “...” **não faz parte da notação**, está aí para reduzir espaço

Derivações

```

expr  ::= expr + digit (p1
      | expr - digit  (p2
      | digit          (p3
digit ::= 0 | 9         (p4

```

- Uma gramática **deriva cadeias de símbolos terminais**:
 - Começando pelo símbolo inicial
 - Substituindo repetidamente a cabeça da produção pelo seu corpo

```

expr
expr + digit
expr - digit + digit
digit - digit + digit
9 - 5 + 2

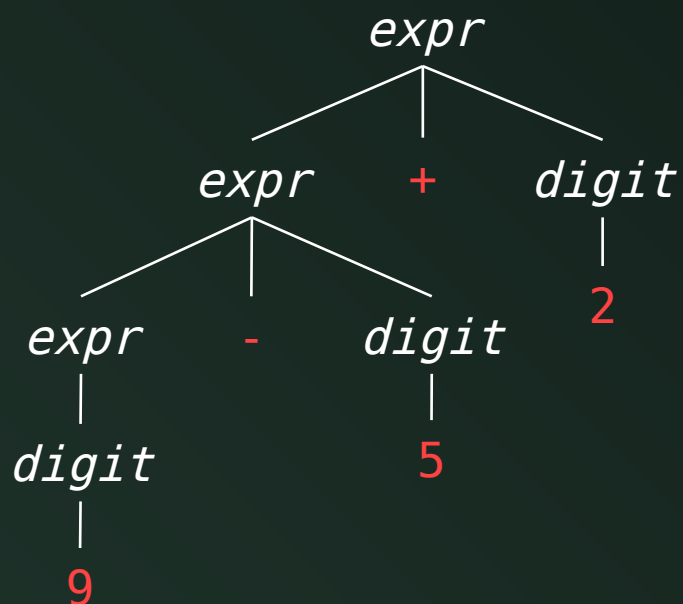
```

9 - 5 + 2 é uma cadeia que faz parte da nossa linguagem porque é possível encontrar uma derivação para ela



- As **cadeias de terminais** que podem ser **derivadas a partir do símbolo inicial** formam a **linguagem** definida pela gramática

Árvores de Derivação

- Uma **árvore de derivação** mostra como o símbolo inicial da gramática deriva uma cadeia da linguagem



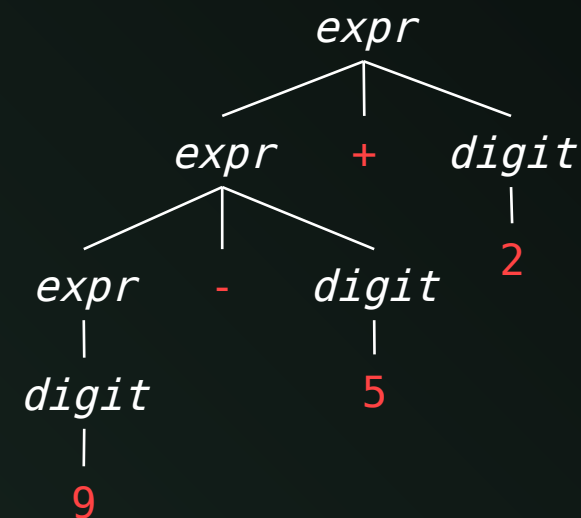
A árvore de derivação
para a cadeia
9-5+2

<i>expr</i>		<i>expr</i> +	(p ₁
		<i>digit</i>)
		<i>expr</i> -	(p ₂
		<i>digit</i>)
		<i>digit</i>	(p ₃
)
<i>digit</i>		0 ... 9	(p ₄
<i>t</i>)

Árvore de Derivação

- Formalmente, uma **árvore de derivação** é uma árvore com as seguintes **propriedades**:

- A **raiz** é rotulada pelo símbolo inicial
- Cada **folha** é rotulada por um terminal ou por ϵ (vazio)
- Cada **nó interior** é rotulado por um não-terminal
- Se A é o não-terminal rotulando um **nó interior** e X_1, X_2, \dots, X_n são seus **filhos**
 - Deve haver uma produção $A \rightarrow X_1 X_2 \dots X_n$
 - Cada X_1, X_2, \dots, X_n deve representar um terminal ou não-terminal
 - Se $A \rightarrow \epsilon$ é uma produção, A pode ter ϵ como seu único filho

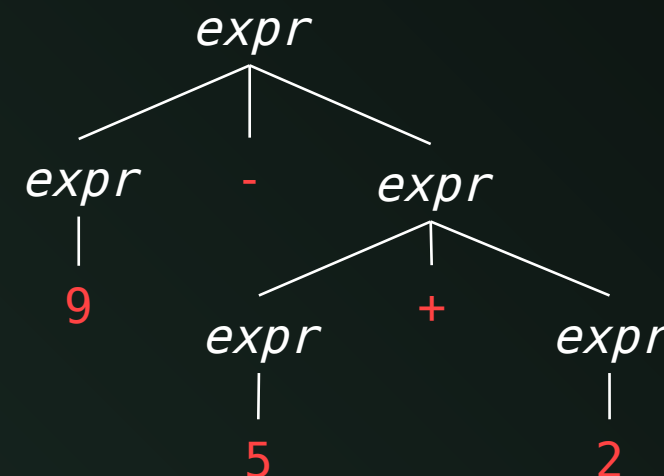
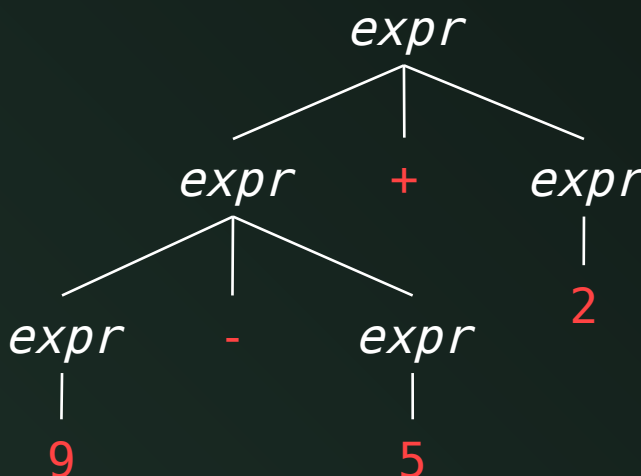


Ambiguidade

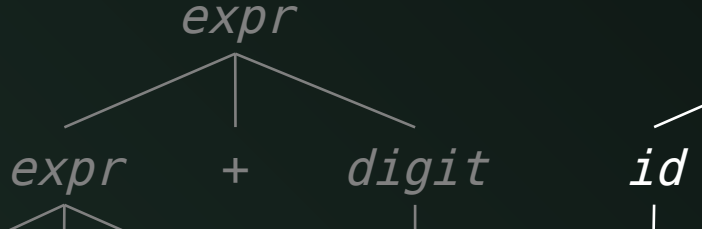
- Quando uma cadeia de terminais pode ser **gerada por mais de uma árvore de derivação**, a gramática é dita **ambígua**
 - Gramáticas ambíguas possuem mais de uma interpretação
 - Um **compilador não consegue trabalhar** com ambiguidade[†]

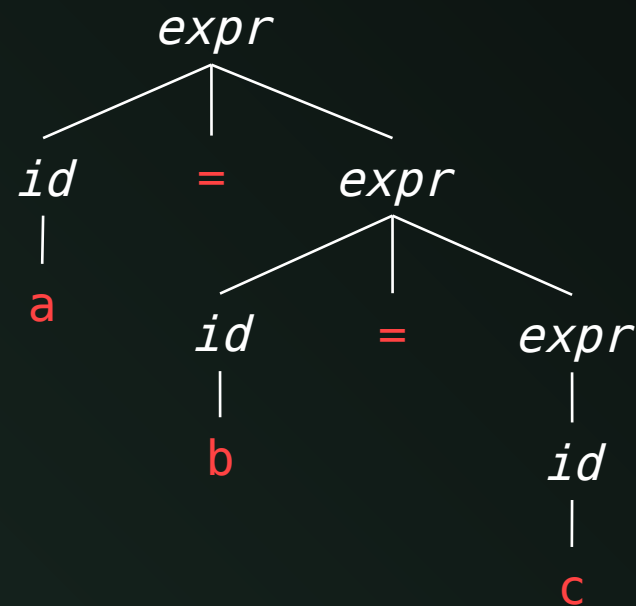
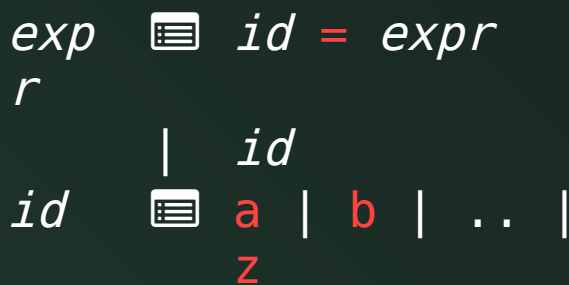
Duas árvores para a cadeia 9-5+2

expr  *expr* + *expr*
| *expr* - *expr*
| 0 | ... | 9



Associatividade de Operadores

- Os operadores possuem associatividade
 - Soma e subtração são operadores associativos à esquerda
 - A expressão $9-5+2$ é interpretada como $(9-5)+2$
 - Atribuição é associativa à direita
 - A expressão $a = b = c$ é interpretada como $a = (b = c)$
- 
- ```
graph TD; expr1[expr] --- expr2[expr]; expr1 --- plus[+]; expr1 --- digit[digit]; expr2 --- id[id]
```









# Precedência de Operadores

- Alguns operadores tem precedência sobre outros
  - Multiplicação e divisão tem precedência sobre soma e subtração
    - A expressão  $9+5*2$  é interpretada como  $9+(5*2)$
  - É necessário separar níveis de precedência na gramática
    - O não-terminal *expr* será usado para soma e subtração
    - O não-terminal *term* será usado para multiplicação e divisão

*expr*   
|  
|

*expr* + *term* *term*   
*expr* - *term* *term* \* *fact*  
*term*  
|  
*term* / *fact*  
|  
*fact*

*fact*   
|  
*digi*  ( *expr* )  
*digi*  0 | ... | 9

# Exercício

1. Considerando a gramática representada pelas produções abaixo, **encontre a derivação** das cadeias:

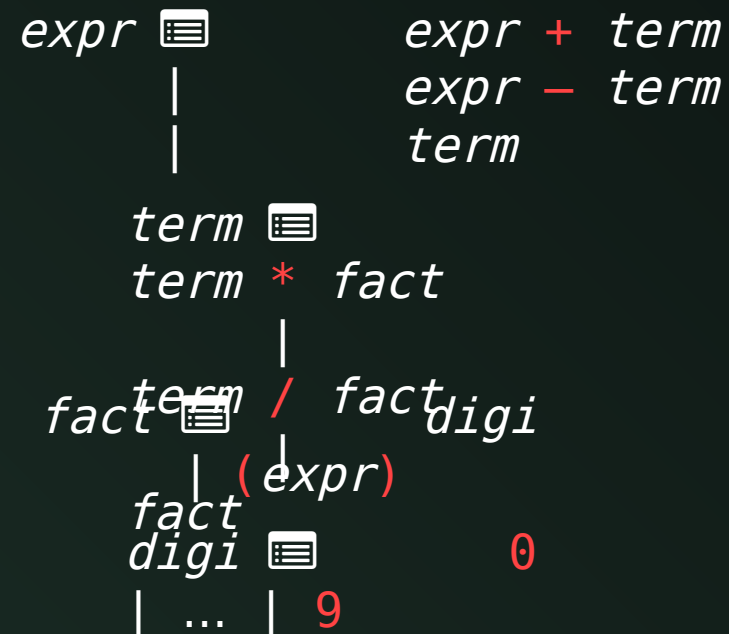
a)  $9 - 5 + 2$

b)  $9 * 3 - 4$

c)  $9 + 6 * 7$

d)  $(5 - 3) / 4$

e)  $((2 + 3) - (5 * 2 + 9 - 7))$

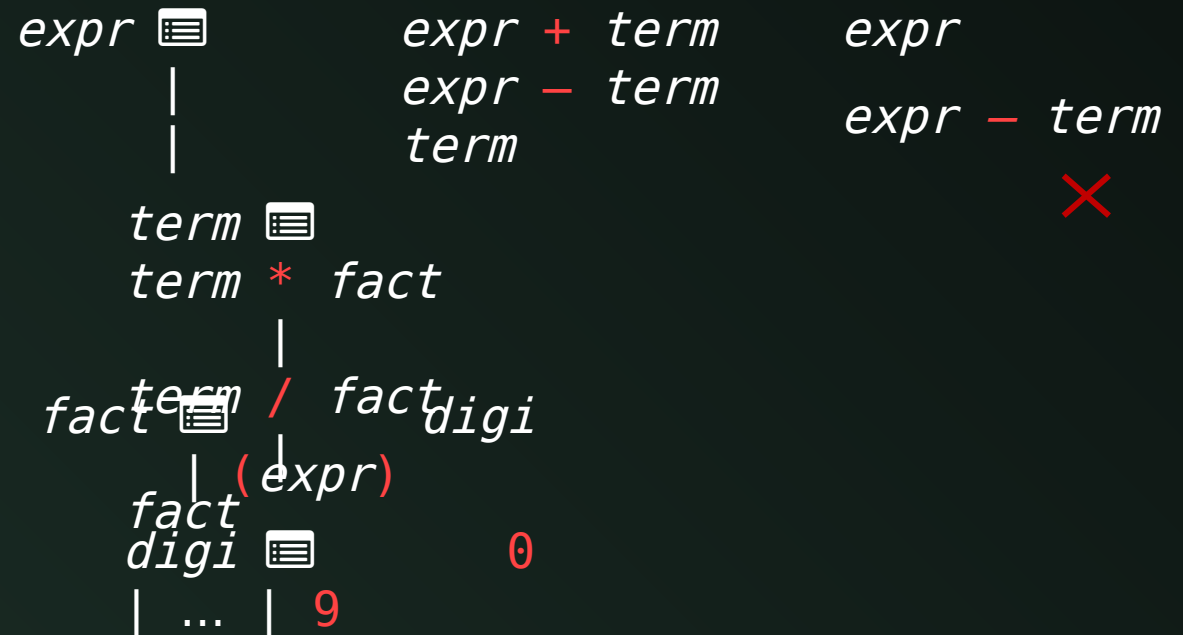


# Exercício

1. Encontre a **derivação** das cadeias:

a) 9-5+2

*expr*  
*expr* + *term*  
*expr* - *term* +  
*term*  
*term* - *term* +  
*term*  
*fact* - *fact* +  
*fact*  
*digi* - *digi* +  
*digi*  
9 - 5 + 2

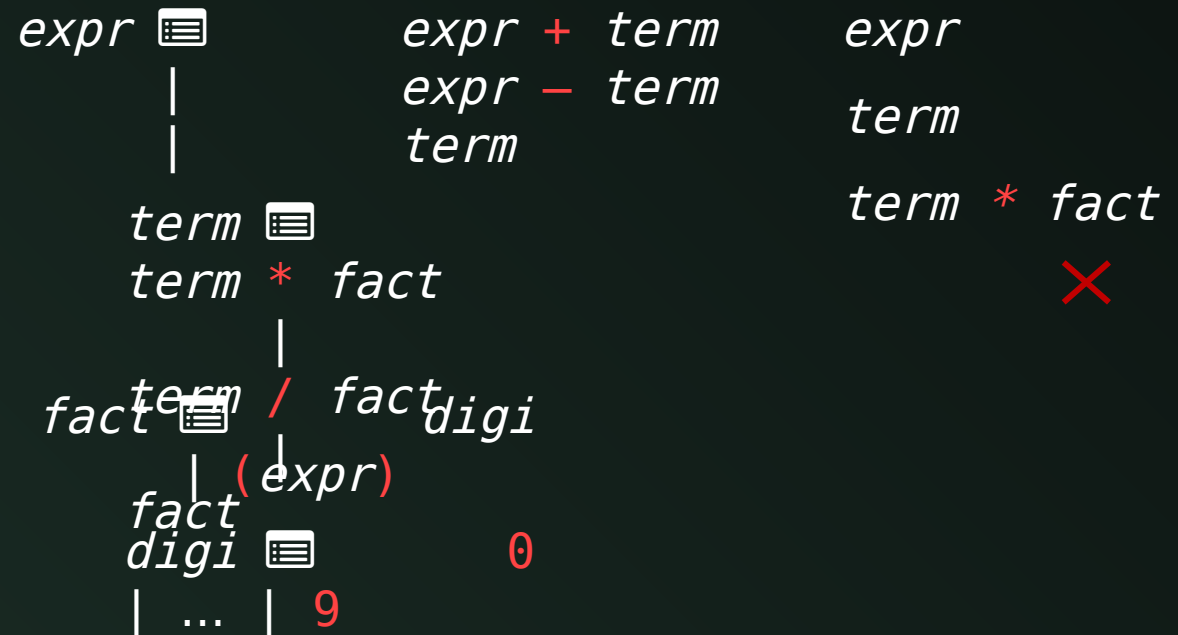


# Exercício

1. Encontre a **derivação** das cadeias:

b)  $9*3-4$

$expr$   
 $expr - term$   
 $term - term$   
 $term * fact -$   
 $term * digi -$   
 $fact * 3 - digi$   
 $9 * 3 - 4$



# Exercício

1. Encontre a **derivação** das cadeias:

c)  $9+6*7$

*expr*

*expr* + *term*

*expr* + *term* \*

*fact*

*term* + *fact* \*

*digi*

*fact* + *digi* \* 7

*digi* + 6 \* 7

9 + 6 \* 7

*expr* 

|  
|

*term* 

*term* \* *fact*

|

*term* / *fact* *digi*

*fact* 

| (*expr*)

*fact*

*digi* 

| ... | 9

*expr* + *term*

*expr* - *term*

*term*

*expr*

*term*

*term* \* *fact*

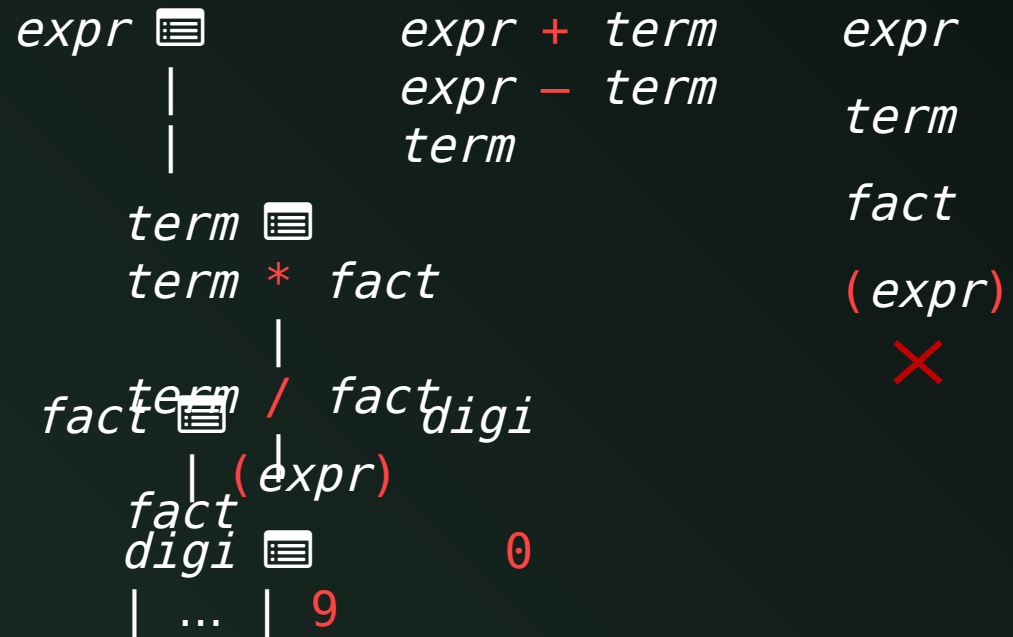
✗

# Exercício

1. Encontre a **derivação** das cadeias:

d)  $(5-3)/4$

*expr*  
*term*  
*term* / *fact*  
*fact* / *digi*  
*(expr)* / 4  
*(expr - term)* / 4  
*(term - fact)* / 4  
*(fact - digi)* / 4  
*(digi - 3)* / 4  
*(5 - 3)* / 4



# Tradução Dirigida por Sintaxe

- A tradução dirigida por sintaxe é obtida **anexando-se regras** ou **fragmentos de código** às produções de uma gramática
  - O que dá origem a dois métodos de tradução:
    - Definição dirigida por sintaxe
    - Esquema dirigido por sintaxe
  - Ambos trabalham com produções da gramática  
 $expr \rightarrow expr_1 + term$  O **subscrito** em  $expr_1$  é usado apenas para **distinguir** a  $expr$  que aparece no **corpo** daquela que aparece na **cabeça** da produção



# Definição Dirigida por Sintaxe

- Uma **definição dirigida por sintaxe** associa:
  - A cada símbolo não-terminal da gramática um **conjunto de atributos**
  - A cada produção um **conjunto de regras**

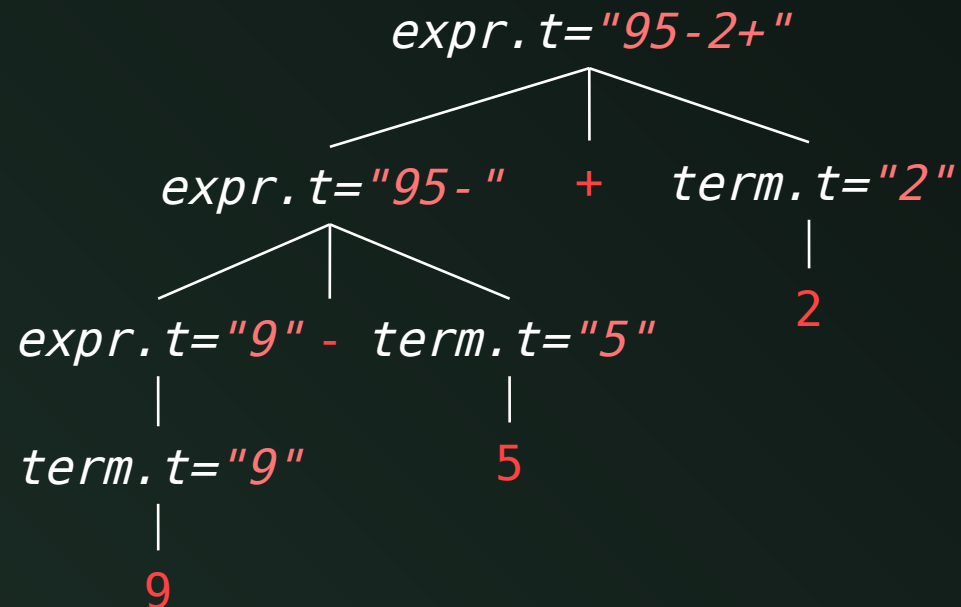
| Gramática                        | Regras Semânticas                                  |
|----------------------------------|----------------------------------------------------|
| $expr \Rightarrow expr_1 + term$ | $expr.t = expr_1.t \parallel term.t \parallel '+'$ |
| $expr \Rightarrow expr_1 - term$ | $expr.t = expr_1.t \parallel term.t \parallel '-'$ |
| $expr \Rightarrow term$          | $expr.t = term.t$                                  |
| $term \Rightarrow 0$             | $term.t = '0'$                                     |
| $term \Rightarrow 1$             | $term.t = '1'$                                     |
| ...                              | ...                                                |
| $term \Rightarrow 9$             | $term.t = '9'$                                     |

O símbolo  $\parallel$  é o **operador de concatenação de cadeias**

# Definição Dirigida por Sintaxe

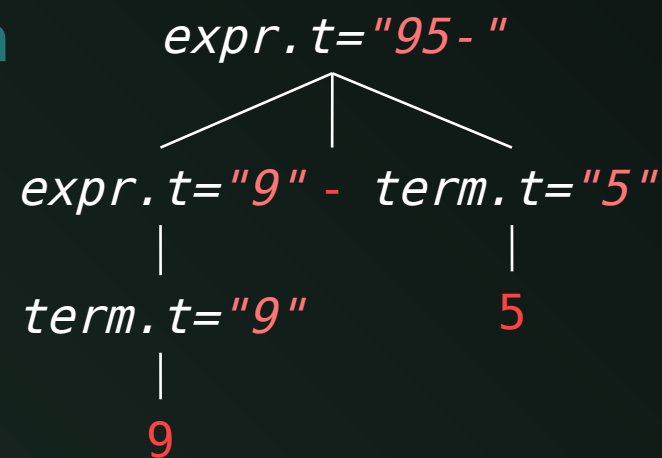
- Para obter a tradução, aplicam-se as regras para avaliar os atributos em cada nó em uma árvore de derivação

Árvore de derivação  
**anotada** para a  
cadeia 9-5+2



# Definição Dirigida por Sintaxe

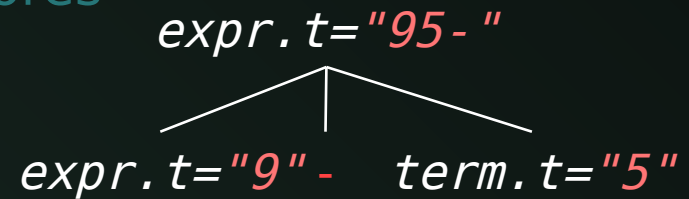
- Um percurso em árvore é usado para avaliar os atributos
  - Podemos usar uma busca em profundidade (depth-first)
    - É um percurso que visita recursivamente os filhos de cada nó
    - Os nós mais profundos (mais distantes da raiz) são visitados primeiro
- A definição dirigida por sintaxe não impõem uma ordem de avaliação dos atributos
  - Qualquer ordem que calcule um atributo depois de todos os outros dos quais ele depende é aceitável



# Definição Dirigida por Sintaxe

- Atributo sintetizado

- Um atributo é considerado sintetizado se o seu valor, em um nó da árvore de derivação, for determinado a partir dos valores dos seus atributos e dos atributos dos seus filhos



- Definição dirigida por sintaxe simples

- Um definição é considerada simples quando a tradução é obtida a partir da concatenação das traduções dos não-terminais no corpo da produção, na mesma ordem da produção

$expr \Rightarrow expr_1 + term \quad | \quad expr.t = expr_1.t \parallel term.t \parallel '+'$

# Esquema Dirigido por Sintaxe

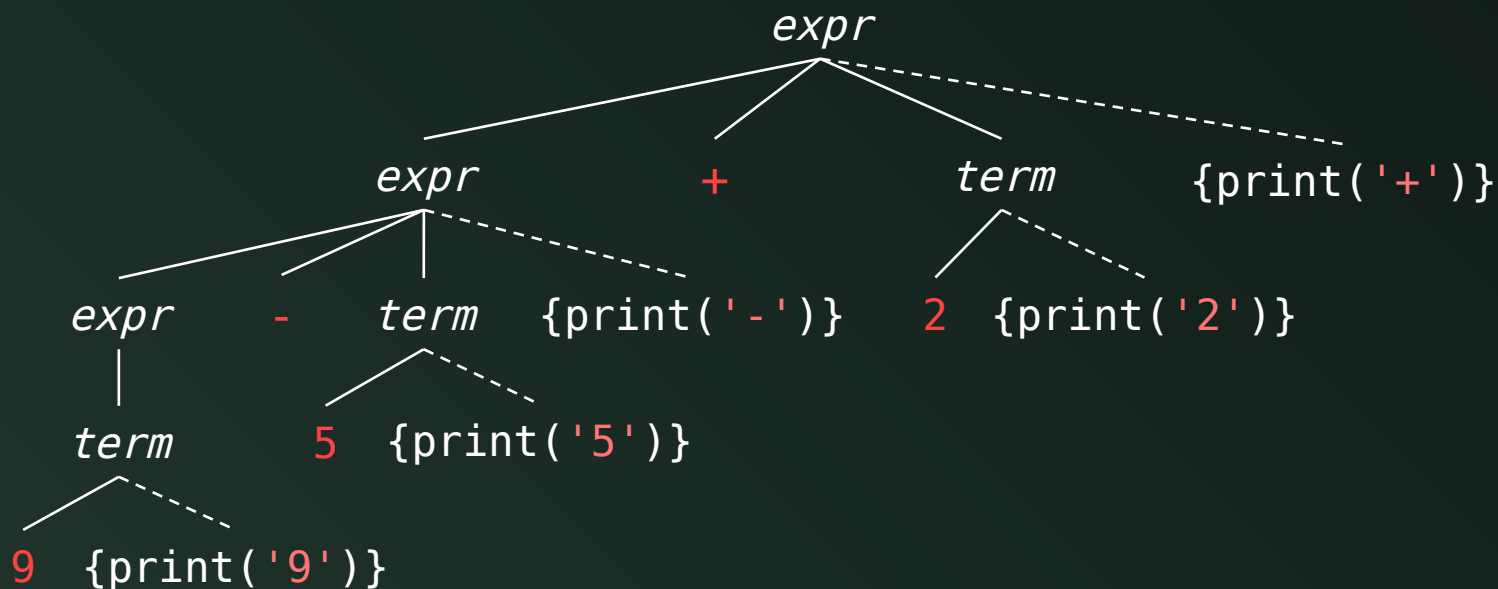
- É outra técnica de tradução
  - Obtém uma tradução **sem armazenar** sequências de caracteres
  - Conecta **fragmentos de código** às produções da gramática
    - Fragmentos são chamados de **ações semânticas**
    - A ação semântica é marcada com { } e inserida na posição de sua execução

```
expr ::= expr1 + term { print('+') }
 | expr1 - term { print('-') }
 | term
```

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

# Esquema Dirigido por Sintaxe

- O esquema de tradução usa a **árvore de derivação**
  - Uma **ação** é indicada por um **filho extra** na árvore
  - A ação é executada na visita de uma **busca em profundidade** (esq-dir)



Árvore de derivação  
com ações semânticas  
para a cadeia `9-5+2`

# Métodos de Tradução

- Os métodos apresentados:
  - Produzem a mesma tradução
  - Necessitam de uma busca em profundidade na árvore de derivação
  - Mas utilizam técnicas diferentes
    - Definição Dirigida por Sintaxe:
      - Concatena sequências de caracteres
      - Armazena atributos dos nós da árvore de derivação
    - Esquema de Tradução Dirigido por Sintaxe:
      - Constrói a tradução de forma incremental sem usar armazenamento
      - Exibe os valores por meio de ações semânticas



# Exercício

1. Construa **árvores de derivação anotadas** para as cadeias em formato pós-fixado:
  - a)  $95 - 2 +$
  - b)  $952 + -$
2. Construa um **esquema de tradução** dirigido por sintaxe que traduza expressões aritméticas (apenas com soma e subtração) da notação pós-fixada para a notação infixada.

# Exercício

- Ambos os exercícios consideram que a entrada será uma **expressão em formato pós-fixado**. É preciso então criar uma **gramática** para representar estas expressões:

```
expr ::= expr expr +
 | expr expr -
 | term
term ::= 0 | ... | 9
```

- Para entender porque a gramática acima funciona, faça os exercícios do Laboratório 03.

# Resumo

- Uma **gramática** é formada por um conjunto de produções
  - Descrevem as construções de uma linguagem
  - Cadeias válidas podem ser visualizadas por árvores de derivação
- A **tradução dirigida por sintaxe** permite construir um tradutor simples usando uma busca em profundidade na árvore de derivação:
  - Anotada com **atributos**
  - Modificada por **ações semânticas**
- Essa técnica de tradução ilustra de forma simples a **análise sintática**