

Compiladores

Trabalho 1

Fellype Siqueira Barroso, Lucas Darlindo Freitas Rodrigues

¹Universidade Federal do Oeste do Pará (UFOPA) – Santarém – PA – Brazil
Instituto de Engenharia e Geociência - IEG

Dada a gramática que gera atribuições (5):

```
ASSIGN ::= LEFT > (EXPR) | LEFT < (EXPR) | LEFT = REST
LEFT  ::= ID | ID[EXPR]
REST  ::= LEFT = REST | (EXPR)
EXPR  ::= EXPR + TERM | EXPR - TERM | TERM
TERM  ::= TERM * UNARY | TERM / UNARY | UNARY
UNARY ::= + UNARY | - UNARY | FACTOR
FACTOR ::= (EXPR) | DIGIT | LEFT
ID     ::= a | b | ... | z
DIGIT  ::= 0 | 1 | ... | 9
```

1. Ajustar a gramática para torná-la LL(1)

```
ASSIGN ::= LEFT ASSIGN'
ASSIGN' ::= > (EXPR) | < (EXPR) | = REST
LEFT  ::= ID LEFT'
LEFT'  ::= [EXPR] | ε
REST  ::= LEFT = REST | (EXPR)
EXPR  ::= TERM EXPR'
EXPR'  ::= + TERM EXPR' | - TERM EXPR' | ε
TERM  ::= UNARY TERM'
TERM'  ::= * UNARY TERM' | / UNARY TERM' | ε
UNARY  ::= + UNARY | - UNARY | FACTOR
FACTOR ::= (EXPR) | DIGIT | LEFT
ID     ::= a | b | ... | z
DIGIT  ::= 0 | 1 | ... | 9
```

2. Apresentar 5 *strings* que são aceitas pela gramática, sendo que juntas elas devem utilizar todas as regras da gramática. E em seguida mostrar a árvore de derivação para cada uma dessas *strings*.

Listing 1. Atribuição a uma variável escalar com operador <

```
a < ( ( a + b ) * ( c - d ) )
```

[illegible]

```
d = ((a / b) * (c * 9) * a[y])
```



```
c[2] = ( (a + b + c) * c[1] )
```


