PROJETO COMPILADORES

(Parte 1 e 2)

Gustavo Mascarenhas Amorim - 12211BCC017
João Caio Pereira Melo - 12211BCC052
Lucas Sabbatini Janot Procopio - 12211BCC019

1. Projeto da Linguagem

1.1. Definição da Gramática livre de contexto

<programa> ::= 'main' ID '()' <bloco>

<bloco> ::= 'inicio' <declaracao_vars> <sequencia_comandos> 'fim'

<declaracao_vars> ::= <lista_ids> '->' <tipo> ';' <declaracao_vars> | ε

<lista_ids> ::= ID | ID ',' <lista_ids>

<tipo> ::= 'int' | 'char' | 'float'

<sequencia_comandos> ::= <comando> <sequencia_comandos> | ε

<comando> ::= <comando_atribuicao> | <comando_selecao> | <comando_repeticao>

comando_atribuicao> ::= ID '=' <expressao> ';'

<comando_selecao> ::= 'caso' '(' <condicao> ')' 'entao' <comando_ou_bloco> <parte_senao>

<parte_senao> ::= 'senao' <comando_ou_bloco> | ε

<comando_ou_bloco> ::= <comando> | <bloco>

<comando_repeticao> ::= 'enquanto' '(' <condicao> ')' 'faca' <comando_ou_bloco> | 'repita' <comando_ou_bloco> 'ate' '(' <condicao> ')' ';'

<condicao> ::= <expressao> <op_relacional> <expressao>

<op_relacional> ::= '==' | '!=' | '<' | '>' | '<=' | '>='

<expressao> ::= <expressao> '+' <termo> | <expressao> '-' <termo> | <termo>

<termo> ::= <termo> '*' <fator> | <termo> '/' <fator> | <termo> '^' <fator> | <fator>

<constante> ::= CONST_INT | CONST_FLOAT | CONST_CHAR

1.2. Identificação dos tokens usados na gramática

A) Palavras Reservadas

Nome do Token	Lexema Exemplo	Tipo do Atributo	Descrição
MAIN	main	N/A	Palavra-chave de início do programa.
INICIO	inicio	N/A	Palavra-chave que marca o início de um bloco.
FIM	fim	N/A	Palavra-chave que marca o final de um bloco
INT	int	N/A	Palavra-chave para o tipo de dado inteiro.
CHAR	char	N/A	Palavra-chave para o tipo de dado caractere.
FLOAT	float	N/A	Palavra-chave para o tipo de dado real.
CASO	caso	N/A	Palavra-chave de início de um comando de seleção.
ENTAO	entao	N/A	Palavra-chave do comando de seleção.
SENAO	senao	N/A	Palavra-chave opcional do comando de seleção.
ENQUANTO	enquanto	N/A	Palavra-chave do comando de repetição.
FACA	faca	N/A	Palavra-chave do comando de repetição enquanto.
REPITA	repita	N/A	Palavra-chave do comando de repetição.
ATE	ate	N/A	Palavra-chave do comando de repetição

B) Identificadores

Nome do Token	Lexema Exemplo	Tipo do Atributo	Descrição
ID	idade, nota	Ponteiro para Tabela de Símbolos	Nome de programas ou variáveis.

C) Constantes

Nome do Token	Lexema Exemplo	Tipo do Atributo	Descrição
CONST_INT	150, 32767	Valor Inteiro	Constante numérica inteira (0 a 32767).
CONST_FLOAT	5.3, 0.1E-2	Valor Real	Constante numérica de ponto flutuante.
CONST_CHAR	'A', 'c'	Valor Caractere	Constante do tipo caractere.

D) Separadores

Nome do Token	Lexema Exemplo	Tipo do Atributo	Descrição
ABRE_PARENT	(	N/A	Abre parênteses.
FECHA_PARENT	)	N/A	Fecha parênteses.
PONTO_VIRGULA	;	N/A	Ponto e vírgula, finalizador de comando.
VIRGULA	,	N/A	Vírgula, para separar identificadores.
SETA	- >	N/A	Seta, para declaração de variáveis.

E) Especiais

Nome do Token	Lexema Exemplo	Tipo do Atributo	Descrição
COMENTARIO	/* ... */	N/A	Bloco de comentário.

1.3. Padrões dos Tokens (Expressões Regulares)

A seguir estão os padrões, definidos por meio de expressões regulares, para o reconhecimento de cada token.

- LETRA: [a-zA-Z]
- DIGITO: [0-9]
- ID: {LETRA}({LETRA}|{DIGITO})*
    - *Observação: As palavras reservadas (main, int, etc.) seguem o padrão de ID, mas são tratadas como tokens específicos.*
- CONST_INT: {DIGITO}+
    - *Observação: A validação do intervalo de 0 a 32767 é realizada na análise semântica.*
- CONST_FLOAT: ({DIGITO}+\.{DIGITO}*|\.{DIGITO}+) ((E|e) [+\-]? {DIGITO}+)?
- CONST_CHAR: '[^'\n\r]'
- OP_ATRIBUICAO: =
- OP_RELACIONAL: == | != | <= | >= | < | >
- OP_ARITMETICO: \+ | \- | * | / | \^
- SEPARADORES: \(| \) | ; | , | ->
- COMENTARIO: /* (.* \n*)* */
- WHITESPACE: [\t\r\n]+ (Ignorado pelo analisador)

2. Análise Léxica

2.1. Diagrama de Transição

A) Palavras Reservadas

Token: MAIN

Expressão Regular: **main**

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → m → a → i → n → Estado Final (Token: MAIN)

Token: INICIO

Expressão Regular: **inicio**

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → i → n → i → c → i → o → Estado Final (Token: INICIO)

Token: FIM

Expressão Regular: `fim`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → f → i → m → Estado Final (Token: FIM)

Token: INT

Expressão Regular: `int`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → i → n → t → Estado Final (Token: INT)

Token: FLOAT

Expressão Regular: `float`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → f → l → o → a → t → Estado Final (Token: FLOAT)

Token: CASO

Expressão Regular: `caso`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → c → a → s → o → Estado Final (Token: CASO)

Token: ENTAO

Expressão Regular: `entao`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → e → n → t → a → o → Estado Final (Token: ENTAO)

Token: ENQUANTO

Expressão Regular: `enquanto`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → e → n → q → u → a → n → t → o → Estado Final (Token: ENQUANTO)

Token: FACA

Expressão Regular: `faca`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → f → a → c → a → Estado Final (Token: FACA)

Token: REPITA

Expressão Regular: `repita`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → r → e → p → i → t → a → Estado Final (Token: REPITA)

Token: ATE

Expressão Regular: `ate`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → a → t → e → Estado Final (Token: ATE)

B) Identificadores

Token: ID

Expressão Regular: `[a-zA-Z]([a-zA-Z0-9])*`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → [a-zA-Z] → Estado Intermediário (S1)
S1 → [a-zA-Z0-9_] (loop no S1)
S1 → Estado Final (Token: ID)

C) Constantes

Token: CONST_INT

Expressão Regular: `[0-9]+`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → [0-9] → Estado Intermediário (S1)
S1 → [0-9] (loop no S1)
S1 → Estado Final (Token: CONST_INT)

Token: CONST_FLOAT

Expressão Regular: `[0-9]+\.[0-9]+([Ee][+-]?[0-9]+)?`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → [0-9] → Estado Intermediário (S1)
S1 → [0-9] (loop no S1)
S1 → . → Estado Intermediário (S2)
S2 → [0-9] → Estado Intermediário (S3)
S3 → [0-9] (loop no S3)
S3 → Estado Final (Token: CONST_FLOAT)
S3 → [Ee] → Estado Intermediário (S4)
S4 → [+-] (opcional) → Estado Intermediário (S5)
S5 → [0-9] → Estado Intermediário (S6)
S6 → [0-9] (loop no S6)
S6 → Estado Final (Token: CONST_FLOAT)

Token: CONST_CHAR

Expressão Regular: `\\\\'[^\\n\\r\\']\\\\'`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → \\\\' → Estado Intermediário (S1)
S1 → Qualquer caractere (exceto \\\\'\\n\\', \\\\'\\r\\', \\\\'\\') → Estado Intermediário (S2)
S2 → \\\\' → Estado Final (Token: CONST_CHAR)

D) Operadores

Token: =

Expressão Regular: `=`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → = → Estado Final (Token: ATRIBUICAO)

Token: !=

Expressão Regular: `!=`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → ! → Estado Intermediário (S1)
S1 → = → Estado Final (Token: DIFERENTE)

Token: <

Expressão Regular: `<`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → < → Estado Final (Token: MENOR_QUE)

Token: >

Expressão Regular: `>`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → > → Estado Final (Token: MAIOR_QUE)

Token: <=

Expressão Regular: `<=`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → < → Estado Intermediário (S1)
S1 → = → Estado Final (Token: MENOR_IGUAL)

Token: >=

Expressão Regular: `>=`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → > → Estado Intermediário (S1)
S1 → = → Estado Final (Token: MAIOR_IGUAL)

Token: +

Expressão Regular: `+`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → + → Estado Final (Token: SOMA)

Token: -

Expressão Regular: `-`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → - → Estado Final (Token: SUBTRACAO)

Token: *

Expressão Regular: `*`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → * → Estado Final (Token: MULTIPLICACAO)

Token: /

Expressão Regular: `/`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → / → Estado Final (Token: DIVISAO)

Token: ^

Expressão Regular: `^`

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → ^ → Estado Final (Token: EXPONENCIACAO)

E) Separadores

Token: ABRE_PARENT

Expressão Regular: (

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → (→ Estado Final (Token: ABRE_PARENT)

Token: FECHA_PARENT

Expressão Regular:)

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) →) → Estado Final (Token: FECHA_PARENT)

Token: PONTO_VIRGULA

Expressão Regular: ;

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → ; → Estado Final (Token: PONTO_VIRGULA)

Token: VIRGULA

Expressão Regular: ,

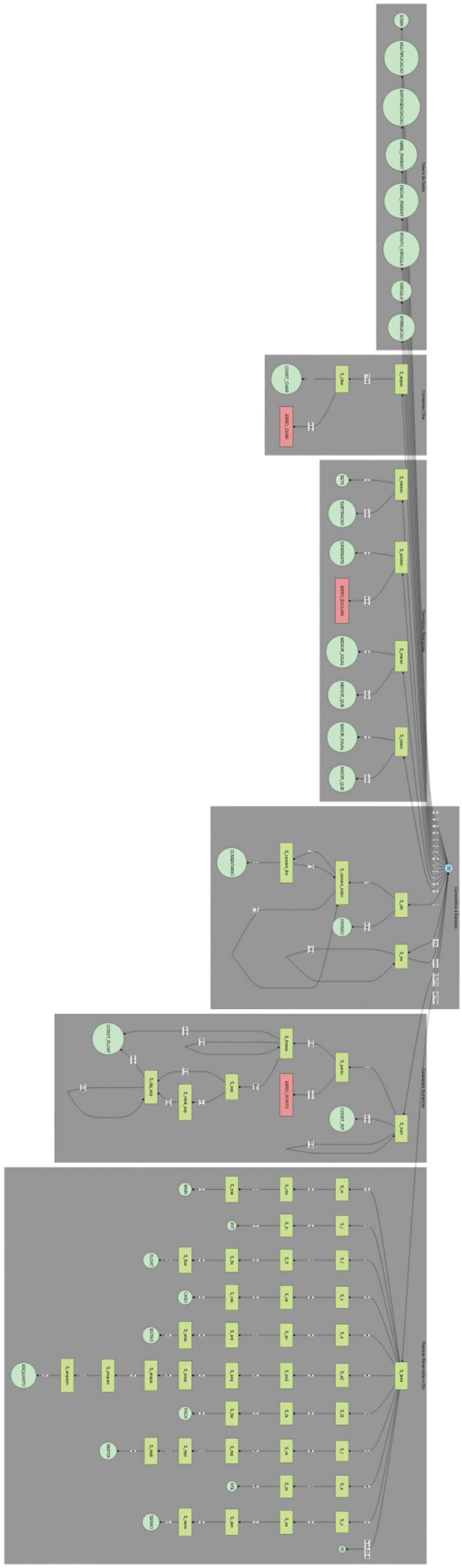
Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → , → Estado Final (Token: VIRGULA)

Token: SETA

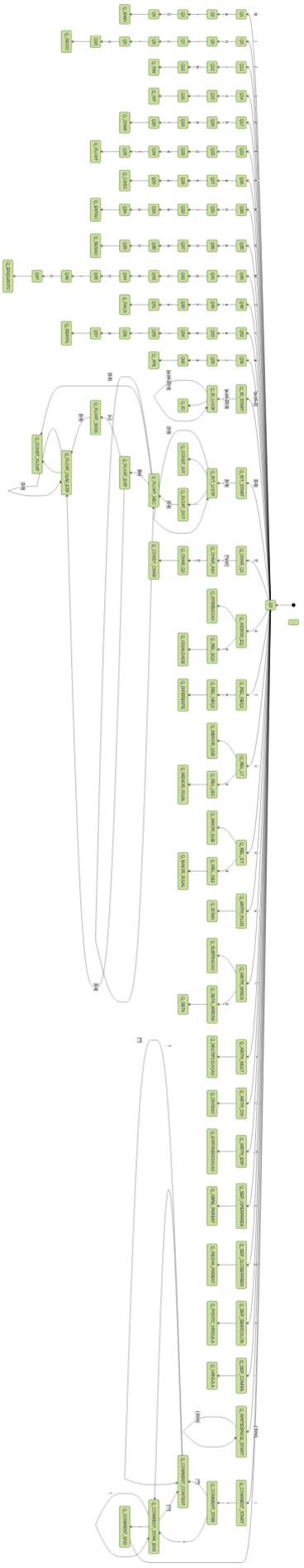
Expressão Regular: →

Diagrama de Transição (NFA/DFA):
Estado Inicial (S0) → - → Estado Intermediário (S1)
S1 → > → Estado Final (Token: SETA)

2.2. Unificar os NFAs individuais em um único NFA global


SOMA
MULTIPLICACAO
EXPONENCIACAO
ABRE_PARENT
FECHA_PARENT
PONTO_VIRGULA
VIRGULA
ATRIBUICAO

2.3. Converter NFA global em um DFA



ESTADO ATUAL	SÍMBOLO ENTRADA	PRÓXIMO ESTADO
Q0	m	Q1
Q1	a	Q2
Q2	i	Q3
Q3	n	Q4 (Token: MAIN)
Q0	i	Q5
Q5	n	Q6
Q6	i	Q7
Q7	c	Q8
Q8	i	Q9
Q9	o	Q10 (Token: INICIO)
Q0	f	Q11
Q11	i	Q12
Q12	m	Q13 (Token: FIM)
Q0	i	Q14
Q14	n	Q15
Q15	t	Q16 (Token: INT)
Q0	c	Q17
Q17	h	Q18
Q18	a	Q19
Q19	r	Q20 (Token: CHAR)
Q0	f	Q21
Q21	l	Q22
Q22	o	Q23
Q23	a	Q24

Q24	**t**	**Q25 (Token: FLOAT)**
Q25	**c**	**Q26**
Q26	**a**	**Q27**
Q27	**s**	**Q28**
Q28	**o**	**Q29 (Token: CASO)**
Q0	**e**	**Q30**
Q30	**n**	**Q31**
Q31	**t**	**Q32**
Q32	**a**	**Q33**
Q33	**o**	**Q34 (Token: ENTAO)**
Q0	**s**	**Q35**
Q35	**e**	**Q36**
Q36	**n**	**Q37**
Q37	**a**	**Q38**
Q38	**o**	**Q39 (Token: SENAO)**
Q0	**e**	**Q40**
Q40	**n**	**Q41**
Q41	**q**	**Q42**
Q42	**u**	**Q43**
Q43	**a**	**Q44**
Q44	**n**	**Q45**
Q45	**t**	**Q46**
Q46	**o**	**Q47 (Token: ENQUANTO)**
Q0	**f**	**Q48**
Q48	**a**	**Q49**
Q49	**c**	**Q50**

Q50	a	Q51 (Token: FACA)
Q0	r	Q52
Q52	e	Q53
Q53	p	Q54
Q54	i	Q55
Q55	t	Q56
Q56	a	Q57 (Token: REPITA)
Q0	a	Q58
Q58	t	Q59
Q59	e	Q60 (Token: ATE)
Q0	[a-zA-Z]	Q61
Q61	[a-zA-Z0-9]	Q61 (Token: ID)
Q0	[0-9]	Q62
Q62	[0-9]	Q62 (Token: CONST_INT)
Q62	.	Q63
Q63	[0-9]	Q64
Q64	[0-9]	Q64 (Token: CONST_FLOAT)
Q64	[Ee]	Q65
Q65	[+-]	Q66
Q66	[0-9]	Q67
Q67 (Token: CONST_FLOAT)	[0-9]	Q67 (Token: CONST_FLOAT)
Q0	\\\\'	Q68
Q68	[^\\n\\r\\']	Q69
Q69	\\\\'	Q70 (Token: CONST_CHAR)

Q0	**=**	**Q71 (Token: ATRIBUICAO)**
Q71	**=**	**Q72 (Token: IGUALDADE)**
Q0	**!**	**Q73**
Q73	**=**	**Q74 (Token: DIFERENTE)**
Q0	**<**	**Q75 (Token: MENOR_QUE)**
Q75	**=**	**Q76 (Token: MENOR_IGUAL)**
Q0	**>**	**Q77 (Token: MAIOR_QUE)**
Q77	**=**	**Q78 (Token: MAIOR_IGUAL)**
Q0	**+**	**Q79 (Token: SOMA)**
Q0	**-**	**Q80 (Token: SUBTRACAO)**
Q80	**>**	**Q81 (Token: SETA)**
Q0	*****	**Q82 (Token: MULTIPLICACAO)**
Q0	**/**	**Q83 (Token: DIVISAO)**
Q0	**^**	**Q84 (Token: EXPONENCIACAO)**
Q0	**(**	**Q85 (Token: ABRE_PARENT)**
Q0	**)**	**Q86 (Token: FECHA_PARENT)**
Q0	**;**	**Q87 (Token: PONTO_VIRGULA)**
Q0	**,**	**Q88 (Token: VIRGULA)**

Q0	[\t\r\n]	Q0 (Ignorar)
Q0	/	Q89
Q89	*	Q90
Q90	[^*]	Q90
Q90	*	Q91
Q91	[^\/]	Q90
Q91	/	Q0 (Ignorar Comentário)