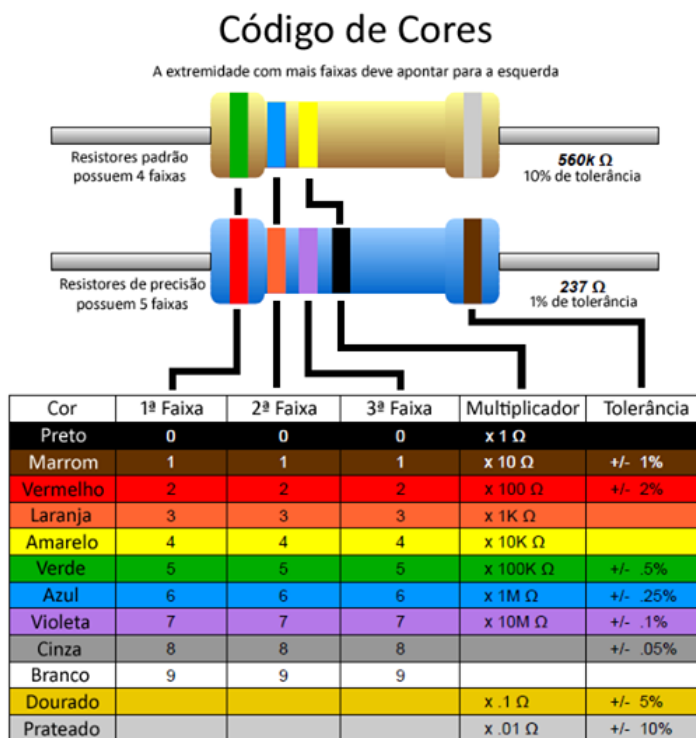


UNIFEI	Universidade Federal de Itajubá Instituto de Engenharia de Sistemas e Tecnologias da Informação-IESTI
Trabalho 01 de Compiladores (ECOM06)	
Bárbara Alves de Paiva Barbosa - 2020003139 Maria Clara Martins Santana - 2020012227 Ryan Wyllyan Ribeiro Inácio - 2020001770	

1. Introdução

Este documento tem como objetivo introduzir o conceito do projeto de desenvolvimento da linguagem ColorOhm. Essa linguagem foi criada com o propósito de simplificar as operações associadas ao uso de resistores. Ela permite a conversão do valor numérico de um resistor para o seu código de cores, conforme ilustrado na Imagem 1. Além disso, a ColorOhm viabiliza a realização de operações fundamentais com resistores, incluindo a conexão em série e em paralelo.

Imagem 1 - Código de cores resistores



Fonte: Apostila Eletrônica Básica IFC. Disponível em:

<https://professor.luzerna.ifc.edu.br/ricardo-kerschbaumer/wp-content/uploads/sites/43/2018/03/Apostila-Eletrônica-Básica-parte-3.pdf>

Portanto, será apresentado neste relatório um exemplo de um possível algoritmo da linguagem aqui projetada. Além disso, pode ser encontrado as expressões regulares, tokens e autômatos relacionados à linguagem.

2. Exemplo do código

```
ohm
//declaração
value r1, r2, r3;
resistor rc, rb, ra, rz;

//atribuição
r1 = 200;
rc = [m,m,r];

//conversão
r2 = (value) [g,r,m]; //conversão para valor
ra = (resistor) 1500; //conversão para resistor
rb = (resistor) r1;

//operações
r3 = r1 : r2; //serie
rz = rc | rb | ra; //paralelo

//imprimindo os valores calculados
show[r1,rc,r2,ra,rb,r3,rz];

endohm
```

3. Expressões Regulares

Em primeiro momento, tem-se abaixo as expressões regulares relacionadas a linguagem. Contudo, ao depender do nível de complexidade e para fins de organização esta seção foi dividida em sub tópicos, os quais agrupam expressões com características semelhantes.

inicio	=	"ohm"
fim	=	"endohm"
terminadorLinha	=	"; "
variavel	=	{letra} ⁺ {digito letra} [*]
valor	=	{digito} ⁺ {digito ⁺ ponto digito ⁺ }
digito	=	{0 – 9}
letra	=	{a – zA – Z}
atribuicao	=	" = "
tipo	=	resistor valorResistor
conversao	=	conversaoValor conversaoResistor

3.1. Operações dos resistores

paralelo	=	" "
série	=	": "
conversaoValor	=	"re2va"
conversaoResistor	=	"va2re"

3.2. Símbolos Especiais

virgula	=	", "
ponto	=	". "
abreColchetes	=	"["
fechaColchetes	=	"]"

3.3. Palavras Reservadas

resistor	=	"resistor"
valorResistor	=	"value"
cor	=	"k" "m" "r" "o" "y" "g" "b" "v" "a" "w"
mostrar	=	"show"

Vale ressaltar que as letras reservadas para cores são correspondentes ao código de cor já apresentado e que a relação de cada letra para cor pode ser observada na Tabela 1.

Tabela 1 - Relação cores e letras reservadas

Letra designada	Cor correspondente
k	Preto
m	Marrom
r	Vermelho
o	Laranja
y	Amarelo
g	Verde
b	Azul
v	Violeta
a	Cinza
w	Branco
d	Dourado
s	Prateado

Fonte: Elaborada pelos autores

3.4. Operações da Linguagem

algoritmo $=$ *inicio {operacoes}⁺ fim*

operacoes $=$
declarando | definicaoValor | definicaoResistor | conversao1 | conversao2 |
conversaoGenerica | operacaoSerie | operacaoParalelo

modeloResistor $=$
abreColchetes cor virgula cor virgula {cor | cor virgula cor} fechaColchetes

declarando $=$
*tipo variavel {virgula variavel} * terminadorLinha*

definicaoValor $=$
variavel atribuicao valor terminadorLinha

definicaoResistor $=$
variavel atribuicao modeloResistor terminadorLinha

conversao1 $=$
variavel atribuicao conversaoValor modeloResistor terminadorLinha

conversao2 $=$
variavel atribuicao conversaoResistor valor terminadorLinha

conversaoGenerica $=$
variavel atribuicao conversao variavel terminadorLinha

operacaoSerie $=$
variavel atribuicao variavel {serie variavel}⁺ terminadorLinha

operacaoParalelo $=$
variavel atribuicao variavel {paralelo variavel}⁺ terminadorLinha

mostrando $=$
*mostrar abreColchetes variavel {virgula variavel} * fechaColchetes*
terminadorLinha

4. Tokens

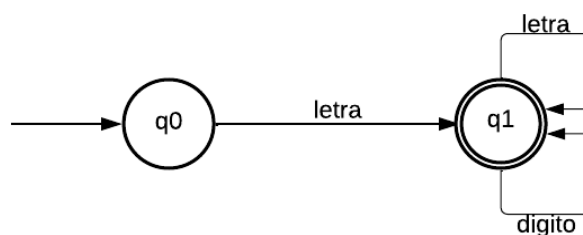
Tabela 2 - Relação dos tokens e suas definições

Definição	Token
variavel	VARIAVEL
valor	VALOR
tipo	TIPO
conversao	CONVERSAO
algoritmo	ALGORITMO
operacoes	OPERACOES
modeloResistor	MODELO_RESISTOR
declarando	DECLARANDO
definicaoValor	DEFINICAO_VALOR
definicaoResistor	DEFINICAO_RESISTOR
conversao1	CONVERSAO1
conversao2	CONVERSAO2
conversaoGenerica	CONVERSAO_GENERICA
operacaoSerie	OPERACAO_SERIE
operacaoParalelo	OPERACAO_PARALELO
mostrando	MOSTRANDO

Fonte: Elaborada pelos autores

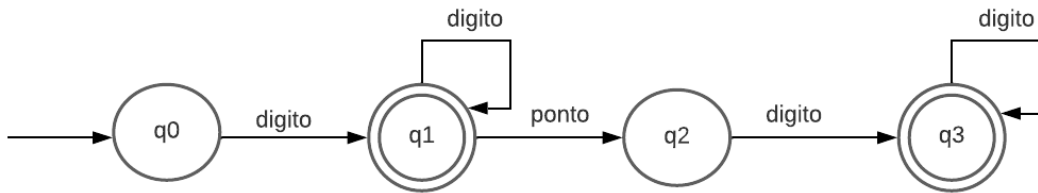
5. Autômatos

Autômato VARIABEL



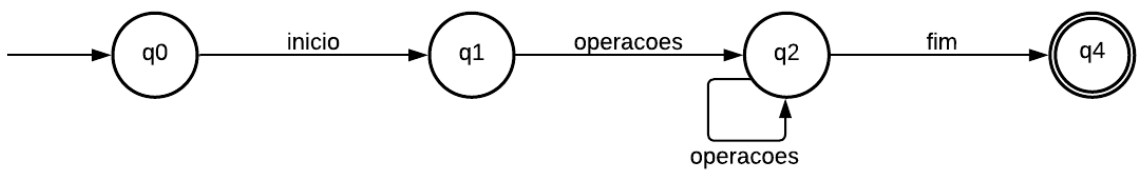
Autômato reconhecedor de variável: precisa iniciar com uma letra. Pode terminar somente com uma letra ou pode vir letras e dígitos depois.

Autômato VALOR



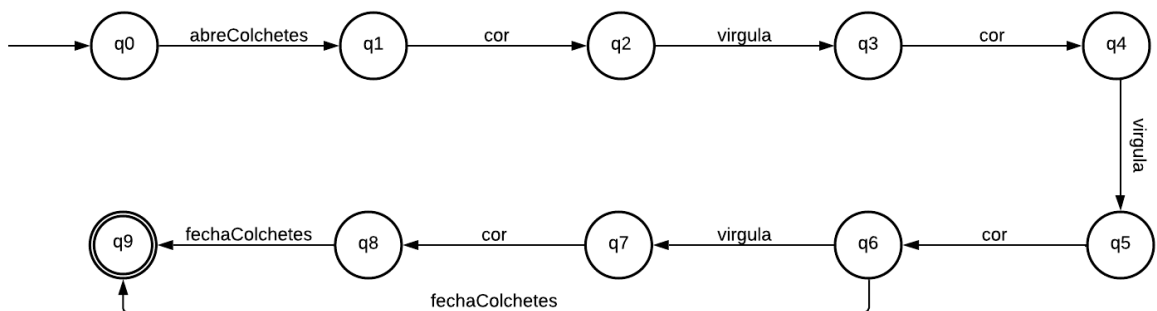
Autômato reconhecedor de valor: pode ser um inteiro com um ou mais dígitos. Ou pode ser fracionário separado por ponto, com um ou mais dígitos após o ponto. Vale ressaltar que este autômato foi desenvolvido para indicar os possíveis valores dos resistores.

Autômato ALGORITMO



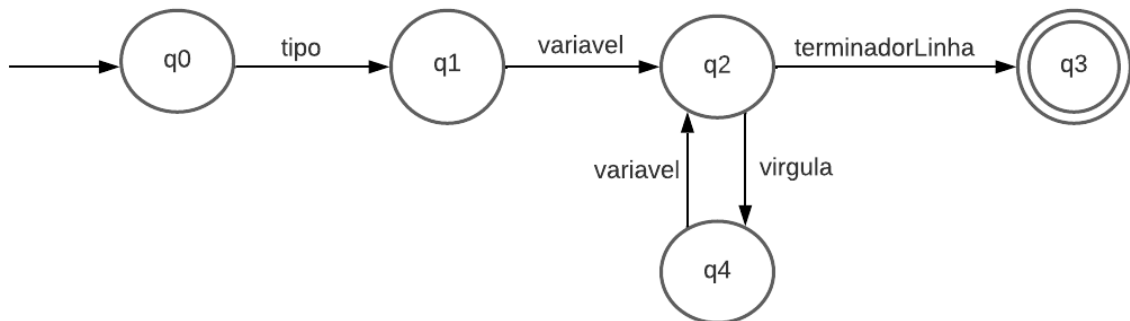
Autômato reconhecedor de algoritmo: precisa reconhecer a palavra reservada para o início e possuir ao menos uma operação. E então pode ter várias operações e ser finalizado.

Autômato MODELO_RESISTOR



Autômato reconhecedor de resistor: o resistor será representado com quatro ou três cores separadas por vírgulas e tudo entre colchetes. Seguindo letras das cores que foram apresentadas na Tabela 1.

Autômato DECLARANDO



Autômato reconhecedor de declaração: a declaração precisa de um tipo e ter ao menos uma variável, podendo ter mais variáveis separadas por vírgula, mas todas serão do mesmo tipo. Ao fim é necessário um terminador de linha para chegar no estado final. Desta forma, este autômato verifica a declaração de todas as variáveis utilizadas nos algoritmos.

Autômato DEFINICAO_VALOR



Autômato reconhecedor de definição de valor: é colocada uma variável que receberá o valor e então o símbolo de atribuição seguido pelo valor que a variável receberá, por fim um terminador de linha para chegar ao estado final.

Autômato DEFINICAO_RESISTOR



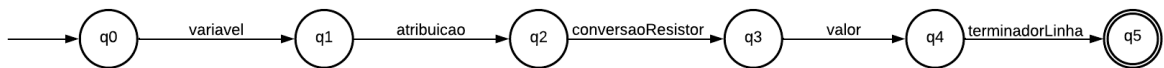
Autômato reconhecedor de definição de resistor: é colocada uma variável que receberá o resistor e então o símbolo de atribuição seguido pelo resistor que a variável receberá, por fim um terminador de linha para chegar ao estado final.

Autômato CONVERSAO1



Autômato reconhecedor de conversão de um resistor para valor: precisa de uma variável que receberá o valor da conversão com um símbolo de atribuição seguido pelo símbolo da conversão para valor e então colocar um valor de cores do resistor para ser convertido. O estado final é atingido com um terminador de linha.

Autômato CONVERSAO2



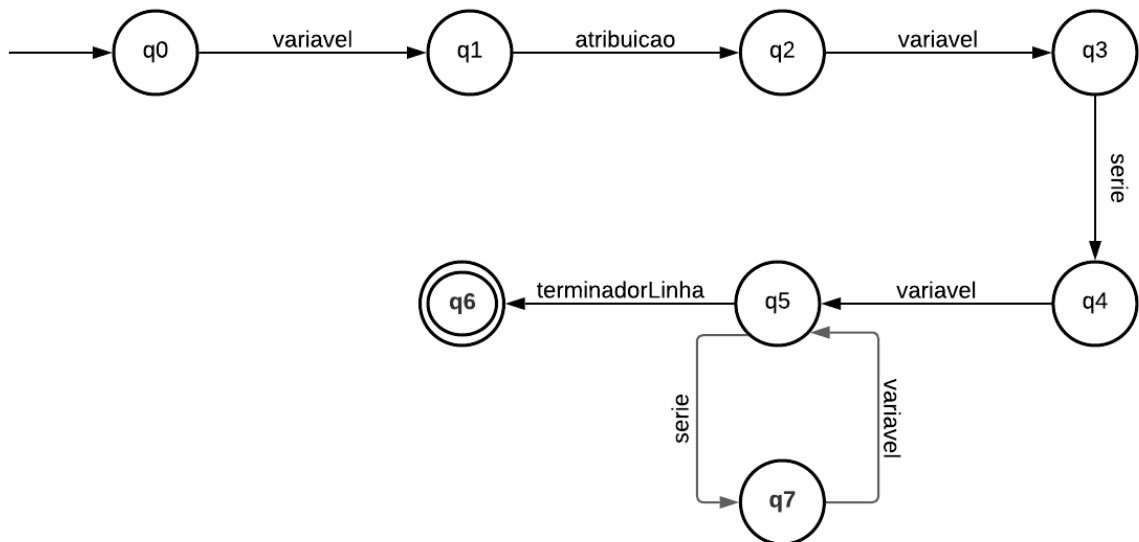
Autômato reconhecedor de conversão de um valor para um resistor: precisa de uma variável que receberá o valor da conversão com um símbolo de atribuição seguido pelo símbolo da conversão para resistor e então colocar o valor a ser convertido. O estado final é atingido com um terminador de linha.

Autômato CONVERSAO_GENERICA



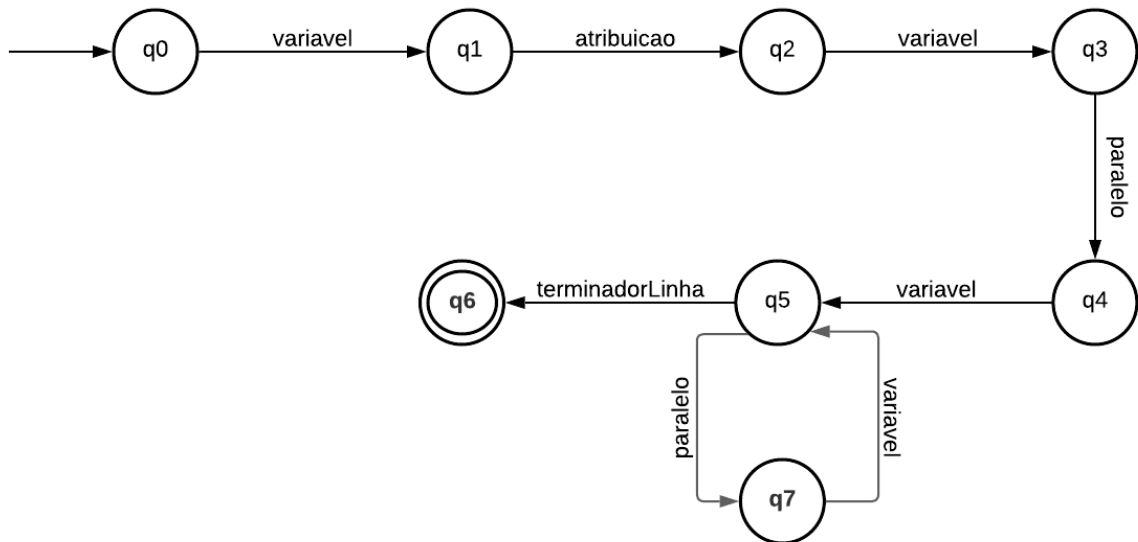
Autômato reconhecedor de conversão genérica: uma conversão consiste em uma variável seguida por uma atribuição com o símbolo da conversão desejada e então colocar a variável com o tipo que precisa ser convertido. O estado final é atingido com um terminador de linha.

Autômato OPERACAO_SERIE



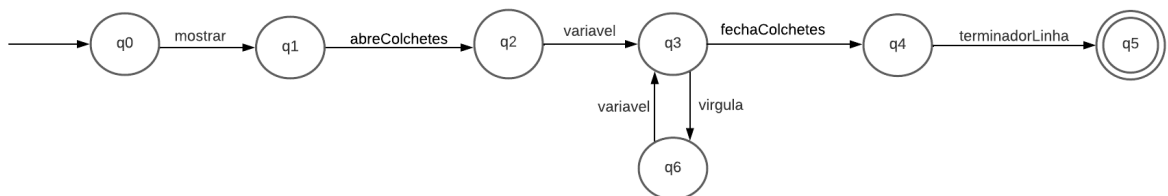
Autômato reconhecedor da operação série de resistor: precisa colocar a variável que receberá o valor da operação, então o símbolo que fará atribuição seguido de pelo menos duas variáveis separadas pelo símbolo da série e em seguida podem vir várias variáveis separadas pela série e por fim um terminador de linha para chegar no estado final.

Autômato OPERACAO_PARALELO



Autômato reconhecedor da operação paralelo de resistor: precisa colocar a variável que receberá o valor da operação, então o símbolo que fará atribuição seguido de pelo menos duas variáveis separadas pelo símbolo da paralelo e em seguida podem vir várias variáveis separadas pelo paralelo e por fim um terminador de linha para chegar no estado final.

Autômato MOSTRANDO



Autômato reconhecedor da impressão de variáveis: exige a palavra reservada que realiza a exibição e precisa abrir colchetes para em seguida reconhecer ao menos uma variável ou várias separadas por vírgula, então os colchetes são fechados e por fim é colocado um terminador de linha.