TLC
Compiladores

Ana Luiza Herrmann Letícia Torres

12 de setembro de 2017

1 Características Relevantes

The Last Code (TLC) é uma linguagem de programação estruturada, imperativa, procedural, inspirada na Saga Star Wars e baseada na linguagem C.

2 Operadores

2.1 Lógicos

A precedência dos operadores lógicos é definida da esquerda para a direita de acordo com a Tabela 1. Não são aceitas operações aritméticas e/ou relacionais combinadas com operações lógicas, apenas variáveis. Caso haja inclusão dos caracteres '(' e ')', a expressão dentro deles possui a maior precedência.

```
jedi (_A + _B red _C > _D) { comandos; } // Nao permitido!
```

```
_X = _A + _B;
_Y = _C > _D;
jedi (_X red _Y) { comandos; } // Correto
```

Tabela 1 – Operadores Lógicos

Descrição	And	Or	Not	Xor
Operador	red	green	blue	violet

2.2 Aritméticos

A ordem de precedência dos operadores aritméticos é da esquerda para a direita, de acordo com a Tabela 2. Caso haja inclusão dos caracteres '(' e ')', a expressão dentro deles possui a maior precedência.

Tabela 2 – Operadores Aritméticos

Descrição	Multiplicação	Divisão	Soma	Subtração	Módulo
Operador	*	/	+	-	%

2.3 Relacionais

Os operadores relacionais não possuem precedência a menos que haja a inserção de parenteses. Não são aceitas operações aritméticas e/ou lógicas combinadas com operações relacionais, apenas variáveis.

Exemplo:

```
jedi (_A + _B == _C + _D) { comandos; } // Nao permitido!
```

```
_X = _A + _B;
_Y = _C + _D;
jedi (_X == _Y) { comandos; } // Correto
```

Tabela 3 – Operadores Relacionais

Descrição	Maior	Maior ou Igual	Menor	Menor ou Igual	Diferença	Igualdade
Operador	>	>=	<	<=	=/=	==

3 Tipos de Dados

Esta linguagem contém declaração de tipos explicita.

Tabela 4 – Tipos de Dados

Descrição	Integer	Char	Float
Tipo	young	master	padawan
Tamanho	4 bytes	1 byte	4 bytes

4 Atribuições

Tabela 5 – Tipos de Atribuições

Atribuição Simples	A = B
Atribuição Composta	A = A + B
Atribuição Unária	count ++

5 Estruturas

5.1 Saltos Condicionais

As estruturas de salto condicional são descritas na Tabela 6.

Tabela 6 – Estruturas de Salto Condicional

Descrição	Se	Senão
Condicional	jedi	sith

5.2 Repetição

As estruturas de repetição são descritas na Tabela 7.

Tabela 7 – Estruturas de Repetição

Descrição	For	While
Laço	republic	yoda

6 Construção de Identificadores

1. A linguagem tem declaração explicita, e todas as variáveis devem ser declaradas com o tipo do dado.
2. O primeiro caractere deve conter somente underscore seguido por letras, e o restante do identificador pode conter também caracteres numéricos.
3. Caracteres especiais não são aceitos.
4. Qualquer identificador deve conter no máximo 15 caracteres.
5. Palavras chaves da linguagem não podem ser utilizadas.
6. A linguagem é *case sensitive*.

7 Palavras Reservadas

Utilizando a Linguagem C como base para a construção da Tabela 8, são listados as palavras reservadas da Linguagem TLC.

Tabela 8 – Palavras Reservadas

Descrição	break	const	continue	return	default
Palavra Reservada	vader	const	stormtrooper	anakin	default

Descrição	sizeof	static	inicio de programa	printf	scanf
Palavra Reservado	sizeof	wookie	enter galaxy	cpo	rtwo

7.1 Entrada e Saída

A estrutura de entrada `rtwo` segue a seguinte formatação: `rtwo < < <entrada do usuário>`.

A estrutura de saída `cpo` segue a seguinte formatação: `cpo > > <mensagem de saída>`.

8 Estrutura Geral da Linguagem

8.1 Estruturação

1. Declaração e definição de variáveis

a) escopo local: inicializada dentro da função principal.

b) variáveis definidas na forma: <tipo>+ <identificador >+ <=>+ <dado>

8.2 Delimitadores

Os delimitadores são descritos na Tabela 7.

Tabela 9 – Delimitadores

Descrição	Bloco	Fim de Linha	Expressões
Delimitador	{ }	;	( )

8.3 Expressões Regulares

8.3.1 Variáveis

Iniciam com um "_"; seguido obrigatoriamente por uma ou mais letras seguidas, de forma opcional, por números.

```
id = _[a-z|A-Z]+[0-9]*
```

8.3.2 Palavras reservadas

Qualquer combinação de letras minúsculas que case com as palavras utilizadas em TLC.

```
p_res = [a-z]+
```

8.3.3 Operadores Relacionais

Combinação de símbolos que case com os Operadores Relacionais.

```
op_rel = [<|>|<=|>=|==|=/=]
```

8.3.4 Operadores Aritméticos

Combinação de símbolos que case com os Operadores Aritméticos.

```
op_arit = [*|/|+|-]
```

8.3.5 Dígitos

Podem ser do tipo Inteiro, `young`; ou Ponto Flutuante, `padawan`. Pontos Flutuantes devem apresentar um Inteiro seguido de um "."e outro Inteiro, mesmo que seja "0".

```
young = [0-9]+
padawan = [0-9]+[.][0-9]+
```

8.3.6 Caractere

Corresponde a um único caractere. Sua representação se dá entre aspas simples (`'<char>'`).

```
master = ['][a-z|A-Z]['] 
```

8.4 Especificação EBNF

```
<digito> := <[0 - 9]>
<letra> := <[a - z] | [A - Z]>
<op_arit> := * | / | + | -
<op_logico> := red | green | blue | violet
<op_rel> := > | < | >= | <= | == | =/=
<op_saida> := > >
<op_entrada> := < <
<atribuicao> := <id> = <stmt>
<stmt> := <term> + <stmt>
        | <term> - <stmt>
        | <term>
<term> := <factor> * <term>
        | <factor> / <term>
        | <factor>
<factor> := <(><stmt><)>
          | <id>
          | <young>
          | <padawan>
          | <master>
<exp_rel> := <id> <op_rel> <exp_rel>
          | <(> <exp_rel> <)>
          | <id>
          | <young>
          | <padawan>
          | <master>
<exp_logica> := <id> <op_logico> <exp_logica>
          | <(> <exp_logica> <)>
          | <id>
          | <young>
          | <padawan>
          | <master>
<atribuicao_unaria> := <id> ++
<stmt_condicao> := jedi(<exp_logica> | <exp_rel>) <{>{<stmt>}<}>
                 | jedi(<exp_logica> | <exp_rel>) <{>{<stmt>}<}>
                        sith <{>{<stmt>}<}>
<republic> := republic <(> {<id> <=> <young>} <;> {<id> <op_rel> <young>}
<;> {<atribuicao_unaria>} <)> <{>{<stmt>}<}>
<yoda> := yoda <(> <id> <op_logico> <id> <)> <{>{<stmt>}<}>
<young> := [<->]{<digito>}+
<master> := [']{<letra>}[']
<padawan>:= [<->] {<digito>}+ "."{<digito>}+
<id> := <underscore> <letra>+ <digito>*
< = > := "="
< =/= > := "=/="
< <= > := "<="
< >= > := ">="
< < > := "<"
```

```
< > > := ">"
<//> := "//"
<*> := "*"
</> := "/"
<-> := "-"
<+> := "+"
<|> := "|"
<%> := "%"
<> := "''
<{> := "{"
<}> := "}"
<++> := "+1";
<negativo> := "-"
<program> := enter galaxy <{> <stmt_lista> <}>
<stmt_lista> := <stmt_condicao>
              | <republic>
              | <yoda>
              | <atribuicao>
```

8.5 Trecho de código

```
enter galaxy
{
    // programa que verifica paridade

    cpo >> "World, Hello! \n";

    young _A;
    young _N = 5;
    young _i;
    young _mod;

    republic (_i = 0; _i < _N; _i++)
    {
        cpo >> "Um numero voce deve digitar: ";
        rtwo << _A;
        _mod = _A % 2;

        jedi (_mod == 0)
        {
            cpo >> "Par _A eh \n";
        } sith {
            cpo >> "Par _A nao eh \n";
        }
    }

    _i = 0;

    yoda (_i == 0)
    {
        cpo >> "Que bom que capaz de compilar ainda nao sou...";
    }

```

```
33        anakin;
34  }
```

8.6 Automato da Linguagem

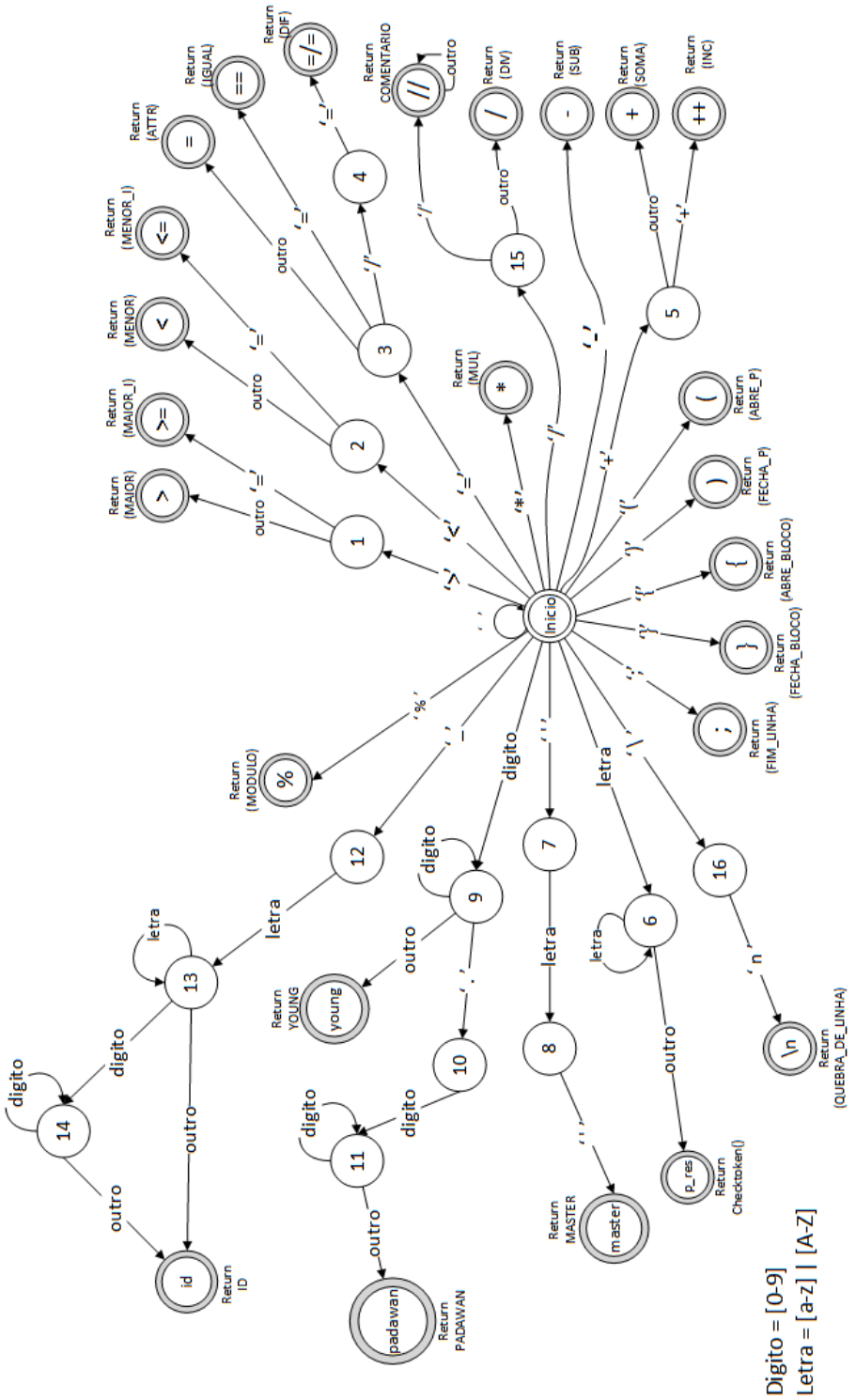

Figura 1 – Autômato da Linguagem TLC