

Analísadores Léxicos e Sintáticos

Projeto Final

Prof. Edward Hermann Haeusler

2019.1

1 Da motivação

A Máquina de Turing é um modelo de computação, ou seja um modelo para representar funções computáveis. É um dos primeiros modelos definidos pelo ser humano. Foi criado por Alan Mathison Turing em 1936 ([1]). Alan Turing demonstrou construtivamente para a comunidade acadêmica deste ano de 1936 a existência de uma máquina programável. Turing definiu matematicamente um modelo de máquina com sintaxe e semântica próprias, e, neste modelo exibiu uma instância de máquina capaz de ler e executar qualquer máquina representada neste próprio modelo. Turing definiu o que hoje chamamos de máquina Universal, i.e., uma máquina U , que ao ler outra máquina qualquer M e um dado ω implementa $M(\omega)$, ou seja a execução de M sobre o dado ω . Em símbolos $U(M, \omega) = M(\omega)$. Tanto M , quanto ω são palavras sobre um certo alfabeto. A própria U pode ser representada como uma palavra, fazendo sentido portanto a execução de U por ela mesma, ou seja, $U(U, \langle M, \omega \rangle)$ que é o mesmo que $U(\langle M, \omega \rangle) = U(M, \omega)$ que resulta na execução de M sobre ω , ou seja, $M(\omega)$. Por conta do seu trabalho de 1936, Alan Mathison Turing é tomado por alguns como sendo o pai da Ciência da Computação. Todos os modelos computacionais conhecidos tem a mesma capacidade computacional (no sentido de realização, não necessariamente de eficiência) da máquina de Turing. Quando se diz que uma linguagem de programação é **Turing-Completa**, isto significa que com ela pode-se implementar qualquer tarefa computável por qualquer outro modelo computacional. Isto pode parecer meio circular, se quiser entender um pouco mais sobre o assunto e o caráter científico do que hoje é cunhado sob o termo “Tese de Turing”¹ leia sobre isso nas referências [2, 3, 5] ou em qualquer livro sobre Teoria da Computação, o que incluí o livro texto da primeira parte do curso.

Com a motivação de dar oportunidade aos alunos de INF1022 de ter algum contato com máquinas de Turing, faremos um compilador de uma linguagem imperativa gerando código objeto como máquinas de Turing. Estas serão posteriormente executadas no ambiente do JFLAP (veja [?]).

2 Do objetivo

O Projeto Final desta disciplina (INF1022) consiste em usar o conjunto de ferramentas Flex/Bison (Lex/Yacc) para geração de compiladores com o objetivo de gerar código de uma linguagem imperativa (Provol-One) em Máquinas de Turing. As máquinas de Turing serão geradas segundo o dialeto XML usado na representação textual das mesmas. Um conjunto de macros, que na terminologia do JFLAP se denomina de BuildingBlocks Turing Machines será fornecido no site da disciplina para auxílio na geração de código. Estas macros servem como uma espécie de **runtime system** para o código gerado. O código gerado pode usar o modelo original de Máquinas de Turing (uma fita somente) ou as máquinas de Turing com mais de uma fita. Fica a critério do grupo a escolha pelo modelo.

¹Informalmente, a tese de Turing diz que toda tarefa é computável, se e somente se, for computada por uma máquina de Turing

De fato o objetivo deste projeto final é fazer com que os alunos da disciplina apliquem os conceitos aprendidos em um projeto mais prático.

3 Do desenvolvimento

A sintaxe da linguagem Provol-One é dada pela gramática abaixo:

```

program  → ENTRADA varlist SAIDA varlist cmds FIM
varlist  → id varlist | id
cmds     → cmd cmds | cmd
cmd      → FACA id VEZES cmds FIM
cmd      → ENQUANTO id FACA cmds FIM
cmd      → SE id ENTAO cmds SENAO cmds | SE id ENTAO cmds
cmd      → id = id | INC(id) | ZERA(id)

```

Todas as variáveis são do tipo número natural (inteiro não-negativo). O comando $Inc(id)$ incrementa em um o conteúdo da variável id , $Zera(id)$ faz o conteúdo da variável id ser 0 (zero), $id = id$ copia o conteúdo da variável a direita na variável da esquerda, ou seja é um comando de atribuição. Os booleanos, usados nos comandos *Enquanto* e de desvio condicional (*Se – Entao* e *Se – Entao – Senao*), são relacionados aos valores numéricos na forma: *falso* é 0 (zero) e *verdadeiro* é qualquer valor positivo. O comando de repetição definido *Faca id vezes < cmds > FIM* repete a execução de *cmds* o número de vezes que for o valor de *id*. Este valor é constante e avaliado no início da execução do comando *Faca*. Assim, um trecho de código na forma *Faca X vezes X = X + 3* será executado o mesmo número de vezes que for o valor de *X* no início da execução do *Faca*. Por exemplo, se *X* tiver falor 5, este será o número de repetições de *cmds* para o trecho acima. Quando *X* tem valor zero, nenhuma execução de *cmds* é realizada. Se houver necessidade de fazer pequenas modificações na sintaxe de Provol-One, fiquem a vontade.

Para facilitar o entendimento do código XML que será gerado, observe a máquina de Turing especificada nas figuras 1 e 2 e seu código XML que no formato “Building Block” é mostrado na figura 3.

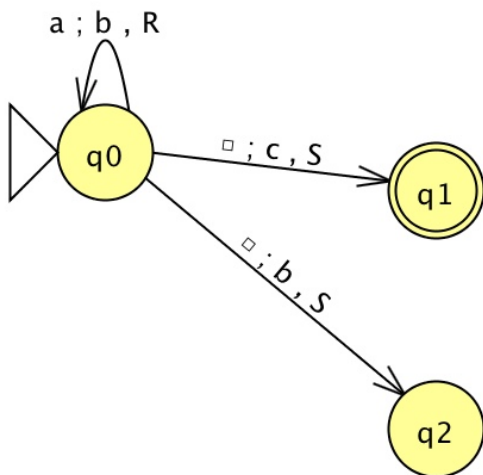


Figura 1: Máquina de Turing usada como bloco na MTda figura 2

4 Entregáveis

A entrega do trabalho constará de:

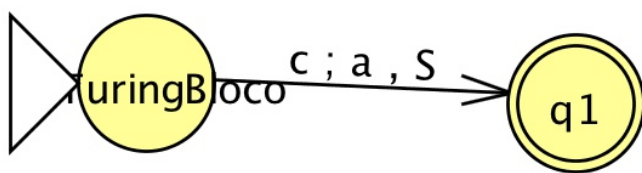


Figura 2: Máquina de Turing com estado (bloco) sendo a MTda figura 1

- (i) arquivos do lex e yacc (flex e bison) usados para gerar o código em JFLAP/XML
- (ii) arquivos de exemplos de uso e execução comparada
- (iii) apresentação do trabalho e entrevista com o grupo.

A entrega do trabalho deve ser feita até o dia 7 de julho (domingo) as 24:00hs. A apresentação do trabalho deve ser feita nos dias 8 e 9 julho com agendamento a partir de 10hs da manhã com 30 minutos para cada grupo.

Referências

- [1] Alan Mathison Turing, On Computable Numbers, with an Application to the Entscheidungsproblem, in *Proceedings of the London Mathematical Society*, n. 42 (2), pp. 230–261, 1936.
- [2] Edward Hermann Haeusler, A celebration of Alan Turing's achievements in the year of his centenary, in *International Transactions on Operational Research*, 19(3), pp 487-491, 2012. <https://doi.org/10.1111/j.1475-3995.2012.00848.x>
- [3] Isabel Cafezeiro e Edward Hermann Haeusler, Computabilidade: Um pouco de História..... um pouco de Matemática, in **VI ERIMG** Escola Regional de Informática MG, 2007. Texto em http://www.bcc.unifal-mg.edu.br/~humberto/disciplinas/2011_1_paa/aulas/complementar_aula01.pdf
- [4] Isabel Cafezeiro, Edward Hermann Haeusler, Henrique Luiz Cukierman e Ivan Costa Marques, *Revista Brasileira de História da Ciência*, Rio de Janeiro, v. 3, n. 2, pp. 231–251, jul-dez 2010
- [5] A. Hodges, Turing: A Natural Philosopher, n. 3, series Great philosophers, 1997, Phoenix Publisher.

```

    <?xml version="1.0" encoding="UTF-8" standalone="no"?><!--Compilado de codigo Provol-One para JFLAP Turing Machines.--><structure>
<type>turing</type>
<automaton>
<!--The list of states.-->
<block id="0" name="M.TuringBloco">
<tag>TesteEditado.jff</tag>
<x>202.0</x>
<y>88.0</y>
<initial/>
</block>
<block id="1" name="q1">
<tag>Machine1</tag>
<x>328.0</x>
<y>94.0</y>
<final/>
</block>
<!--The list of transitions.-->
<transition>
<from>0</from>
<to>1</to>
<read>c</read>
<write>a</write>
<move>S</move>
</transition>
<!--The list of automata-->
<Machine1/>
<TesteEditado.jff>
<!--The list of states.-->
<block id="0" name="q0">
<tag>Machine0</tag>
<x>187.0</x>
<y>122.0</y>
<initial/>
</block>
<block id="1" name="q1">
<tag>Machine1</tag>
<x>326.0</x>
<y>138.0</y>
<final/>
</block>
<block id="2" name="q2">
<tag>Machine2</tag>
<x>326.0</x>
<y>238.0</y>
</block>
<!--The list of transitions.-->
<transition>
<from>0</from>
<to>0</to>
<read>a</read>
<write>b</write>
<move>R</move>
</transition>
<transition>
<from>0</from>
<to>2</to>
<read>d</read>
<write>b</write>
<move>S</move>
</transition>
<transition>
<from>0</from>
<to>1</to>
<read/>
<write>c</write>
<move>S</move>
</transition>
<!--The list of automata-->
<Machine0/>
<Machine2/>
<Machine1/>
</TesteEditado.jff>
</automaton>
    </structure>

```

Figura 3: Código em XML da máquina com Building Block da figura 2