Relatório 1 - Máquina de Turing

Gabriel de Oliveira Almeida Gustavo Lopes Santana João Victor Menezes

1 Introdução

Na disciplina de teoria da computação foi proposto elaborar uma ferramenta junto a esse respectivo relatório, que diz a respeito de visualização e validação de uma máquina de Turing representadas. A ferramenta será desenvolvida em um site Web, na qual será escrita em *javascript*, junto a API *JsPlumb Toolkit*.

2 Máquina de Turing

No projeto separamos os arquivos .js, em um arquivo para comunicação entre a máquina e a biblioteca utilizada para a representação gráfica deles que é o delegateAFND, e outro arquivo para implementação e percorrimento da fita.

```
1        this.transitions = {};
2        this.startState = useDefaults ? 'start' : null;
3        this.acceptStates = useDefaults ? ['accept'] : [];
4        this.history;
```

No trecho de código acima mostra as variáveis que serão utilizadas.

- Transitions é um objeto que será guardado o estado referente, os caracteres da transição e o estado que está conectado.
- startState é uma variável que guarda o estado inicial.
- acceptStates é um vetor que guarda os estados finais da máquina.
- history é onde está guardado a fita

Função que adiciona a transição de estados. Recebe o estado inicial, o caractere a ser escrito, o caractere a ser lido, a direção da fita e o estado final da transição.

```
1        if (!this.transitions[stateA]) {
2      this.transitions[stateA] = {};
3    }
```

```
    if (!this.transitions[stateA][inputRead]) {
      this.transitions[stateA][inputRead] = [];
    }
    this.transitions[stateA][inputRead].push({
      final: stateB,
      write: inputWrite,
      direction: inputRLS
    });
```

A função stepInit pega uma cópia da fita para realizar as transições na máquina de Turing.

```
 AFND.prototype.stepInit = function (input) {
    this.i=1;
    console.log("Executando Turing Machine '" + input + "'");
    var hist = new HistoryLog(input.split(""));
    this.history = this.step(hist, this.startState);
    console.log("RESULT: " + this.history.found);
    return this.history.found;
};
```

2.1 Máquina de Turing para XML

Foi utilizado a estrutura XML que é o objeto em Javascript.
Essa primeira parte é o cabeçalho do XML que o JFlap usa.

```
        var parser = new DOMParser()
        var xml = parser.parseFromString('<?xml version="1.0" encoding="utf
           -8" standalone="no"?><structure></structure>', "application/xml"
           );

        var newElement

        newElement = xml.createElement("type"); //cria um novo node
        xml.getElementsByTagName("structure")[0].appendChild(newElement);//
           aplica o novo node criado em um outro
        xml.getElementsByTagName("type")[0].appendChild(xml.createTextNode(
           'turing')); //atributo em um node

        newElement = xml.createElement("automaton");
xml.getElementsByTagName("structure")[0].appendChild(newElement);
```

Depois vem a parte da tag ¡states¿

```
        $.each(model.states, function (state) {

                if (state === 'q0') {
                        model.states[state].top = 55;
```

```
                        model.states[state].left = 55;
                        model.states[state].startState = true;
                };
                newElement = xml.createElement("state");
                newElement.setAttribute("id", state.slice(1));
                newElement.setAttribute("name", state);
                xml.getElementsByTagName("automaton")[0].appendChild(
                   newElement);

                newElement = xml.createElement("x");
                newElement.appendChild(xml.createTextNode(model.states[
                   state].top + i * 51));
                xml.getElementsByTagName("state")[i].appendChild(newElement
                   );
                newElement = xml.createElement("y");
                newElement.appendChild(xml.createTextNode(model.states[
                   state].left + i * 71));
                xml.getElementsByTagName("state")[i].appendChild(newElement
                   );

                if (model.states[state].isAccept) {
                        newElement = xml.createElement("final"); //  final
                        xml.getElementsByTagName("state")[i].appendChild(
                           newElement);
                } else if (model.states[state].startState) {
                        newElement = xml.createElement("initial"); //
                           initial
                        xml.getElementsByTagName("state")[i].appendChild(
                           newElement);

                }
                i++;
        });
```

E por fim as transições.

```
        $.each(model.transitions, function (index) {

                newElement = xml.createElement("transition");
                xml.getElementsByTagName("automaton")[0].appendChild(
                   newElement);

                newElement = xml.createElement("from");
                newElement.appendChild(xml.createTextNode(model.transitions
                   [index]['stateA'].slice(1)));
                xml.getElementsByTagName("transition")[i].appendChild(
                   newElement);
```

```
9
10                  newElement = xml.createElement("to");
11                  newElement.appendChild(xml.createTextNode(model.transitions
                        [index]['stateB'].slice(1)));
12                  xml.getElementsByTagName("transition")[i].appendChild(
                        newElement);
13
14                  newElement = xml.createElement("read");
15                  if (model.transitions[index]['read'] == emptyLabel) {
16                          newElement.appendChild(xml.createTextNode(model.
                                transitions[index]['read']));
17                          xml.getElementsByTagName("transition")[i].
                                appendChild(newElement);
18                  }
19                  newElement = xml.createElement("write");
20                  if (model.transitions[index]['write'] == emptyLabel) {
21                          newElement.appendChild(xml.createTextNode(model.
                                transitions[index]['write']));
22                          xml.getElementsByTagName("transition")[i].
                                appendChild(newElement);
23                  }
24                  newElement = xml.createElement("move");
25                  newElement.appendChild(xml.createTextNode(model.transitions
                        [index]['direction']));
26                  xml.getElementsByTagName("transition")[i].appendChild(
                        newElement);
27                  i++;
28
29  });
```

2.2 XML para Máquina de Turing

Utilizamos de novo a estrutura XML, onde uma variável recebe esse objeto e percorremos esse objeto para criar a máquina.

```
1   function convertJSON(xml) {
2
3           transitions = {};
4           acceptStates = [];
5
6           var node = xml.getElementsByTagName("type")[0];
7           node.childNodes[0].nodeValue;
8
9           for (var i = 0; i < xml.getElementsByTagName("state").length; i++)
                {
10
```

```
                node = xml.getElementsByTagName("state")[i];

                if (node.getElementsByTagName("initial")[0]) {
                        startState = ('q' + node.getAttribute("id"));
                };
                if (node.getElementsByTagName("final")[0]) {
                        acceptStates.push('q' + node.getAttribute("id"));
                };
        };

        for (var i = 0; i < xml.getElementsByTagName("transition").length;
           i++) {

                node = xml.getElementsByTagName("transition")[i];

                var stateA = 'q' + node.getElementsByTagName("from")[0].
                   childNodes[0].nodeValue;
                var stateB = 'q' + node.getElementsByTagName("to")[0].
                   childNodes[0].nodeValue;
                var inputRead = (node.getElementsByTagName("read")[0].
                   childNodes.length) ?
                node.getElementsByTagName("read")[0].childNodes[0].
                   nodeValue : emptyLabel;
                var inputWrite = (node.getElementsByTagName("write")[0].
                   childNodes.length) ?
                        node.getElementsByTagName("write")[0].childNodes
                           [0].nodeValue : emptyLabel;
                var inputRLS = node.getElementsByTagName("move")[0].
                   childNodes[0].nodeValue;

                if (!this.transitions[stateA]) { this.transitions[stateA] =
                    {}; }
                if (!this.transitions[stateA][inputRead]) { this.
                   transitions[stateA][inputRead] = []; }
                this.transitions[stateA][inputRead].push({ final: stateB,
                   write: inputWrite, direction: inputRLS });

        };

return (serializeJSON());
}
```