

Trabalho de Linguagens Formais - Documentação

Gustavo Azevedo (m91999)

O manipulador de gramáticas foi desenvolvido em Java Script e utilizou das seguintes ferramentas: Bootstrap, Pure CSS e JQuery. A interface foi desenvolvida em HTML e CSS.

1. Interface

Segue abaixo uma imagem do layout da interface.

Trabalho de Linguagens Formais
Gustavo Azevedo

Gramática {

não-terminais

terminais

produção

símbolo inicial

}

G = {(N), (T), P, S)

P = {

}

Acionar

Tipo de Gramática

Senteças Geradas = {

}

Nessa segunda imagem, temos o layout completo, após terem sido escritos os campos e rodado o programa.

Trabalho de Linguagens Formais
Gustavo Azevedo

Gramática {

AS

as

$S \rightarrow aA \mid sA \mid sS$
 $A \rightarrow s \mid a \mid sS$

S

}

G = ({A, S}, {a, s}, P, S)
P = {
 $S \rightarrow aA \mid sA \mid sS$
 $A \rightarrow s \mid a \mid sS$
}

Acionar
Gramática Regular

Senteças Geradas = {
 $S \rightarrow aA \rightarrow asS \rightarrow asaA \rightarrow asaa$
 $S \rightarrow sS \rightarrow ssA \rightarrow sss$
 $S \rightarrow aA \rightarrow asS \rightarrow asaA \rightarrow asaa$
}

| # | aA | sA | sS | s | a | sS |
|---|----|----|----|---|---|----|
| S | A | A | S | - | - | S |
| A | - | - | S | ε | ε | S |

No quadrante superior esquerdo, se encontra as áreas de input para se inserir os dados da gramática.

Trabalho de Linguagens Formais
Gustavo Azevedo

Gramática {

não-terminais

terminais

produção

símbolo inicial

}

G = ({N}, {T}, P, S)
P = {
}
}

Acionar
Tipo de Gramática

Senteças Geradas = {
}
}

Na parte superior do quadrante superior direito, se encontra a escrita da gramática inserida.

Trabalho de Linguagens Formais
Gustavo Azevedo

Gramática {

AS

as

$S \rightarrow aA \mid sA \mid sS$
 $A \rightarrow s \mid a \mid sS$

S

$G = (\{A, S\}, \{a, s\}, P, S)$
 $P = \{$
 $S \rightarrow aA \mid sA \mid sS$
 $A \rightarrow s \mid a \mid sS$
 $\}$

Acionar

Gramática Regular

Senteças Geradas = {
 $S \rightarrow aA \rightarrow asS \rightarrow asaA \rightarrow asaa$
 $S \rightarrow sS \rightarrow ssA \rightarrow sss$
 $S \rightarrow aA \rightarrow asS \rightarrow asaA \rightarrow asaa$
 $\}$

| # | aA | sA | sS | s | a | sS |
|---|----|----|----|------------|------------|----|
| S | A | A | S | - | - | S |
| A | - | - | S | ϵ | ϵ | S |

Pode-se observar que no quadrante superior direito, na área inferior, indica qual o tipo de gramática que ele reconheceu.

Trabalho de Linguagens Formais
Gustavo Azevedo

Gramática {

AS

as

$S \rightarrow aA \mid sA \mid sS$
 $A \rightarrow s \mid a \mid sS$

S

$G = (\{A, S\}, \{a, s\}, P, S)$
 $P = \{$
 $S \rightarrow aA \mid sA \mid sS$
 $A \rightarrow s \mid a \mid sS$
 $\}$

Acionar

Gramática Regular

Senteças Geradas = {
 $S \rightarrow aA \rightarrow asS \rightarrow asaA \rightarrow asaa$
 $S \rightarrow sS \rightarrow ssA \rightarrow sss$
 $S \rightarrow aA \rightarrow asS \rightarrow asaA \rightarrow asaa$
 $\}$

| # | aA | sA | sS | s | a | sS |
|---|----|----|----|------------|------------|----|
| S | A | A | S | - | - | S |
| A | - | - | S | ϵ | ϵ | S |

No quadrante inferior da esquerda, se encontra três sentenças randomicamente geradas para a gramatica inserida.

Trabalho de Linguagens Formais

Gustavo Azevedo

Gramática {

AS

as

S > aA | sA | sS

A > s | a | sS

S

}

G = ({A, S}, {a, s}, P, S)

P = {

S → aA | sA | sS

A → s | a | sS

}

Acionar

Gramática Regular

Senteças Geradas = {

S → aA → asS → asaA → asaa

S → sS → ssA → sss

S → aA → asS → asaA → asaa

}

| # | aA | sA | sS | s | a | sS |
|---|----|----|----|---|---|----|
| S | A | A | S | - | - | S |
| A | - | - | S | ε | ε | S |

E por final, no quadrante inferior direito, se encontra a tabela do autômato finito para a gramática inserida.

Trabalho de Linguagens Formais

Gustavo Azevedo

Gramática {

AS

as

S > aA | sA | sS

A > s | a | sS

S

}

G = ({A, S}, {a, s}, P, S)

P = {

S → aA | sA | sS

A → s | a | sS

}

Acionar

Gramática Regular

Senteças Geradas = {

S → sS → ssS → sssS → ssssA → sssss

S → sS → ssA → sssS → sssaA → sssasS → sssassA → sssasss

S → aA → asS → assS → assaA → assaa

}

| # | aA | sA | sS | s | a |
|---|----|----|----|---|---|
| S | A | A | S | - | - |
| A | - | - | S | ε | ε |

2. Funcionamento

Embaixo segue o código fonte da criar a gramática sendo inserida, ele acontece automaticamente a mediada que os dados são inseridos.

```

//Esses sets pegam os dados escritos nas caixas a
//esquerda, processam eles,
//e escrevem a gramática na caixa da direita
function setNT(nter) {
  let aux = '';
  let aws = nter.split('');
  for (let key in aws) {
    key = aws[key];
    if (/[A-Z]/g.test(key)) {
      aux += key + ', ';
    }
  }
  $('#n1').html(aux.slice(0, -2));
}

function setTer(ter) {
  let aux = '';
  let aws = ter.split('');
  for (let key in aws) {
    key = aws[key];
    if (/[a-z]/g.test(key)) {
      aux += key + ', ';
    }
  }
  $('#t1').html(aux.slice(0, -2));
}

function setSI(si) {
  $('#s1').html(si);
  $('#si').val(si);
}

function setProd(prod) {
  prod = prod.replace(/>/g, '→');
  prod = prod.replace(/&/g, 'ε');
  prod = prod.replace(/(?:\r\n|\r|\n)/g, '<br>');
  $('#p1').html(prod);
}

```

Abaixo se encontra o código para reconhecer o tipo de gramática, que roda quando o botão é clicado.

```
function runProgram() {
  //teste para descobrir o tipo de gramática
  let [gr, glc, gsc, gi] = new Array(4).fill(true);

  //limpa os dados para mais facilmente poder utiliza-los no futuro
  let prod = $('#prod').val();
  prod = prod.replace(/ /g, '');
  let linhas = prod.split('\n');
  let esquerda = [];
  let direita = [];
  let dirSimbolo = [];
  for (const key of linhas) {
    let aux = key.split('>');
    direita.push(aux[1]);
    esquerda.push(aux[0]);
  }
  for (const key of direita) {
    let aux = key.split('|');
    for (const i of aux) {
      dirSimbolo.push(i);
    }
  }
  //fim da limpeza dos dados

  //Procura se possui terminais na esquerda, se sim, não é GR nem GLC
  if (/([a-z])(.*>)/g.test(prod)) {
    gr = false;
    glc = false;
  }

  //Olha o lado esquerdo, se tiver mais de um caractere, não é GR nem GLC
  for (const key in esquerda) {
    for (const iterator of esquerda[key]) {
      if (iterator.length > 1) {
        gr = false;
        glc = false;
      }
    }
  }

  //Procura no lado direito se possui um terminal sozinho ou um terminal seguido de NT,
  //se fugir dessa regra, não é GR.

  //Procura também por vazios, se encontrar, não é GLC nem GSC
  for (const iter of dirSimbolo) {
    for (const key in iter) {
      if (
        /[A-Z]/g.test(iter.charAt(key)) &&
        !/[a-z]/g.test(iter.charAt(key - 1))
      ) {
        gr = false;
      } else if (
        /[a-z]/g.test(iter.charAt(key)) &&
        iter.charAt(key + 1) == '' &&
        iter.charAt(key - 1) != ''
      ) {
        gr = false;
      }
    }
  }
  if (iter.includes('&')) {
    glc = false;
    gsc = false;
  }

  //testa se o lado esquerdo possui o mesmo tamanho ou menor que o lado direito,
  //se não for, não é GSC
  if (gsc) {
    for (const key of linhas) {
      let [esq, dir] = key.split('>');
      let aux = dir.split('|');
      for (const i of aux) {
        if (i.length < esq.length) {
          gsc = false;
          break;
        }
      }
    }
  }

  //Testa se tem um NT na esquerda, se não tiver, não é uma produção válida
  for (const iterator of esquerda) {
    if (!/[A-Z]/g.test(iterator)) {
      gi = false;
      gsc = false;
      gr = false;
      glc = false;
    }
  }
}
```

Abaixo se encontra o código para escrever as sentenças, para fazer tal, se utilizou uma função recursiva.

```

//Início da criação da sentença
//Pega o valor inicial
let inicio = $('#si').val();
let sentenca = inicio;
let sentencas = '';
for (const key in linhas) {
  if (esquerda[key] == inicio) {
    //Repete o processo três vezes
    for (let a = 0; a < 3; a++) {
      //Lança a função para criar a sentença com
      //os dados (inicio, opções da direita do inicio, sentença anterior)
      criaSentenca(esquerda[key], direita[key], esquerda[key]);
      sentenca.replace(/&/g, ''); //Retira o vazío
      sentencas = sentencas + sentenca + '<br>'; //Concatena as sentenças geradas
      sentenca = inicio; //re-inicia o loop
    }
    //escreve na tela a resposta
    sentencas = `Sentenças Geradas = { <br />
      ${sentencas}
    }`;
    $('#resultGramaticas').html(sentencas);
    break;
  }
}

function criaSentenca(nt, t, anterior) {
  let aux = t.split('|');
  let limit = aux.length;
  //Pega uma opção da direita randomicamente e substitui
  let randT = aux[Math.floor(Math.random() * limit)];
  let nova = anterior.replace(nt, randT);
  sentenca = sentenca + ' → ' + nova;
  try {
    //testa se ainda existem NT na sentença, caso existirem,
    //escolha uma randomicamente e repete o processo
    if (/[A-Z]/g.test(nova)) {
      let NT = [];
      for (const key in esquerda) {
        if (nova.includes(esquerda[key])) {
          NT.push(key);
        }
      }
      if (NT.length != 0) {
        let rand = Math.floor(Math.random() * NT.length);
        criaSentenca(esquerda[NT[rand]], direita[NT[rand]], nova);
      } else {
        alert('Erro na produção: Não possui fim');
      }
    }
  } catch (e) {
    alert('Erro na produção: Loop infinito');
  }
}

```

E por fim, está o código para gerar a tabela de autômato finito.

```
// Automato Finito
if (gr) {
  //cria a tabela
  $('#tabela').show();
  let tableHead = `<tr><th scope="col">#</th>`;
  let tableBody = `<tr>`;
  //pega o alfabeto
  let alfabeto = new Set(dirSimbolo);
  for (const i of alfabeto) {
    tableHead += `<th scope="col">${i}</th>`;
  }
  for (const i of linhas) {
    tableBody += `<th class="tableRow" scope="row">${i.split('>')[0]}</th>`;
    for (const a of alfabeto) {
      //pega os conjuntos de estados e testa eles contra o alfabeto
      let regex = new RegExp(`\\b${a}\\b`, 'g');
      if (regex.test(i)) {
        if (/[A-Z]/g.test(a)) {
          let aux = a.replace(/[a-z]/g, '').replace(/(?!^)(?!$)/g, '/');
          tableBody += `<td>${aux}</td>`;
        } else {
          tableBody += `<td>ε</td>`;
        }
      } else {
        tableBody += `<td>-</td>`;
      }
    }
    tableBody += `</tr>`;
  }
  tableHead += `</tr>`;
  //escreve na tela
  $('#tableHead').html(tableHead);
  $('#tableBody').html(tableBody);
} else {
  $('#tabela').hide();
}
```