XML2AA: geração automática de autômatos adaptativos a partir de especificações XML

P. R. M. Cereda e J. José Neto

***Abstract*—Este artigo apresenta uma ferramenta para geração automática de autômatos adaptativos, utilizando a linguagem Java, a partir de especificações XML, de forma consistente e aderente à teoria. Aspectos técnicos do desenvolvimento da ferramenta são apresentados e discutidos, incluindo modos de execução e exemplos de especificações de autômatos, em formato XML, para reconhecimento de linguagens regulares, livres de contexto e dependentes de contexto.**

***Palavras-chave:*—Autômato adaptativo, linguagens de programação, bibliotecas, adaptatividade.**

I. Introdução

Este artigo apresenta uma ferramenta chamada XML2AA para geração automática de autômatos adaptativos, utilizando a linguagem Java, a partir de especificações XML, de forma consistente e aderente à teoria original proposta por José Neto [1]. Espera-se que esta ferramenta ofereça subsídios para a incorporação de elementos adaptativos em programas através de uma interface simplificada, além de atuar como material didático no ensino de autômatos adaptativos.

A organização deste artigo é a seguinte: a Seção II apresenta uma breve revisão bibliográfica da teoria. A Seção III apresenta aspectos técnicos da ferramenta, incluindo seus modos de operação, e discussões sobre implementação. Exemplos de especificações de autômatos, em formato XML, e execução no terminal de consulta integrado da ferramenta são contemplados na Seção IV. As considerações finais são apresentadas na Seção V.

II. Conceitos iniciais

O autômato adaptativo [2], [1] é um extensão do formalismo do autômato de pilha estruturado [1] que permite o reconhecimento de linguagens recursivamente enumeráveis. O termo *adaptativo*, neste contexto, pode ser definido como a capacidade de um dispositivo em alterar seu comportamento de forma espontânea. Um autômato adaptativo, portanto, tem como característica a possibilidade de alterar sua própria topologia durante o processo de reconhecimento de uma dada cadeia, em resposta a algum estímulo [3]. A possibilidade de alteração do autômato ocorre através da utilização de ações adaptativas, que lidam com situações esperadas, mas ainda não consideradas, detectadas na cadeia submetida para reconhecimento pelo autômato [4].

Definição 1 (espaço de autômatos)**.** Ao executar uma transição que contém uma ação adaptativa associada, o autômato sofre mudanças, obtendo-se uma nova configuração. Para a aceitação de uma determinada cadeia, o autômato percorrerá um caminho em um espaço de autômatos; haverá, portanto, um autômato E_0 que iniciará o reconhecimento de uma cadeia w, autômatos intermediários E_i que serão criados ao longo do reconhecimento, e um autômato final E_n que corresponde ao final do reconhecimento de w. Em outras palavras, seja a cadeia $w = \alpha_0\alpha_1 \ldots \alpha_n$; o autômato M descreverá um caminho de autômatos $\langle E_0, \alpha_0\rangle \rightarrow \langle E_1, \alpha_1\rangle \rightarrow \ldots \rightarrow \langle E_n, \alpha_n\rangle$, onde E_i representa um autômato correspondente à aceitação da subcadeia a_i. □

Definição 2 (autômato adaptativo)**.** Um autômato adaptativo M é definido como $M = (Q, A, \Sigma, \Gamma, P, Z_0, q_0, F, E, \Phi)$, onde Q é o conjunto finito não-vazio de estados, A é um conjunto de submáquinas, definidas a seguir, Σ é o alfabeto do autômato, correspondendo ao conjunto finito não vazio dos símbolos de entrada, Γ é o conjunto finito não-vazio de símbolos de pilha, armazenados na memória auxiliar do autômato, P é a relação de transição de estados, $q_0 \in Q$ é o estado inicial (da primeira submáquina), Z_0 é o símbolo marcador de pilha vazia, $F \subseteq Q$ é o conjunto dos estados de aceitação do autômato (da primeira submáquina), E representa o modelo de autômato utilizado (inicialmente, o reconhecimento inicia-se no autômato E_0 do caminho de autômatos), e Φ é o conjunto de funções adaptativas, aplicáveis às transições [1], [3]. □

Definição 3 (submáquina do autômato adaptativo)**.** Uma submáquina $a_i \in A$ do autômato adaptativo M é definida como $a_i = (Q_i, \Sigma_i, P_i, q_{i,0}, F_i, \Phi_i)$, onde $Q_i \subseteq Q$ é o conjunto de estados de a_i, $\Sigma_i \subseteq \Sigma$ é o conjunto de símbolos de entrada de a_i, $q_{i,0}$ é o estado de entrada da sub-máquina a_i, $P_i \subseteq P$ é a relação de transição de estados de a_i, $F_i \subseteq F$ é o conjunto de estados de retorno da sub-máquina a_i, e $\Phi_i \subseteq \Phi$ é o conjunto de funções adaptativas aplicáveis. □

Definição 4 (relação de transição do autômato adaptativo)**.** A relação de transição P é definida como $P \subseteq (\Gamma \times Q \times \Sigma \times (\Phi \cup \{\epsilon\})) \times (\Gamma \times Q \times \Sigma \times (\Phi \times \{\epsilon\}))$, na forma $(\gamma g, e, s\alpha), \mathcal{B} \rightarrow (\gamma g', e', s'\alpha), \mathcal{A}$, onde e, e' são os estados corrente e de destino, repectivamente, s é o símbolo consumido, α é o restante da cadeia de entrada, g é o topo da pilha, g' é o novo topo da pilha, γ é o restante da pilha, $\mathcal{B}$ é uma função adaptativa anterior, e $\mathcal{A}$ é uma função adaptativa posterior. Uma configuração para um autômato adaptativo é um elemento de $E \times Q \times \Sigma^* \times \Gamma^*$, e uma relação entre configurações sucessivas $\vdash$ é definida como:

– *Consumo de símbolo:* $(E_i, q, \sigma w, uv) \vdash (E_i, p, \sigma' w, xv)$, com $p, q \in Q$, $u, x \in \Gamma$, $v \in \Gamma^*$, $\sigma, \sigma' \in \Sigma \cup \{\epsilon\}$, $w \in \Sigma^*$, se σ foi consumido pelo autômato, $x = u$, e

Os autores podem ser contatados através dos seguintes endereços de correio eletrônico: paulo.cereda@usp.br e jjneto@usp.br.

$(\gamma, q, \sigma\alpha) \rightarrow (\gamma, p, \sigma'\alpha) \in P$.

- *Chamada de sub-máquina:* $(E_i, q, w, uv) \vdash (E_i, r, w, xv)$, com $q, r \in Q$, $u \in \Gamma$, $v, x \in \Gamma^*$, $w \in \Sigma^*$, $x = pu$, com chamada da sub-máquina R, estado inicial r, retorno em p, e $(\gamma, q, \alpha) \rightarrow (\gamma p, r, \alpha) \in P$.
- *Retorno de sub-máquina:* $(E_i, q, w, uv) \vdash (E_i, p, w, v)$, com $p, q \in Q$, $u, x \in \Gamma$, $v \in \Gamma^*$, $w \in \Sigma^*$, $u = p$, com retorno de sub-máquina para p, e $(\gamma g, q, \alpha) \rightarrow (\gamma, g, \alpha) \in P$.
- *Funções adaptativas:* $(E_i, q, \sigma w, uv) \vdash^{\mathcal{B},\mathcal{A}} (E_{i+2}, p, \sigma' w, u'v)$, indicando $(E_i, q, \sigma w, uv) \rightarrow^{\mathcal{B}} (E_{i+1}, q, \sigma w, uv) \vdash (E_{i+1}, p, \sigma' w, u'v) \rightarrow^{\mathcal{A}} (E_{i+2}, p, \sigma' w, u'v)$, com $p, q \in Q$, $u, u' \in \Gamma$, $v \in \Gamma^*$, $w \in \Sigma^*$, se $(\gamma u, q, \sigma\alpha), \mathcal{B} \rightarrow (\gamma u, p, \sigma'\alpha), \mathcal{A} \in P$. □

Definição 5 (chamada de função adaptativa)**.** Uma chamada de função adaptativa $\mathcal{F}_i$ assume a forma $\mathcal{F}_i(\tau_{i,1}, \tau_{i,2}, \ldots, \tau_{i,m})$, onde cada $\tau_{i,j}$ representa um argumento passado à função. A declaração de uma função adaptativa $\mathcal{F}_i$ com m parâmetros consiste de um cabeçalho da forma $\mathcal{F}_i(\Theta_{i,1}, \Theta_{i,2}, \ldots, \Theta_{i,m})$ e um corpo contendo nomes (uma lista de identificadores que representam objetos no escopo da função, incluindo variáveis e geradores) e ações (lista de ações adaptativas elementares precedida por uma ação adaptativa inicial e seguida por outra final). A inicialização de uma função adaptativa preenche os geradores e os parâmetros passados e, a seguir, as ações são efetivamente aplicadas. □

Definição 6 (ações adaptativas elementares)**.** São definidos três tipos de ações adaptativas elementares que realizam testes no conjunto de regras ou modificam regras existentes (relação de transição de estados P), a saber:

1) *ação adaptativa elementar de inspeção:* a ação não modifica o conjunto de regras, mas permite a inspeção deste para a verificação de regras que obedeçam um certo padrão.
2) *ação adaptativa elementar de remoção:* a ação remove regras que correspondem a um determinado padrão do conjunto corrente de regras.
3) *ação adaptativa elementar de inclusão:* a ação insere uma regra que corresponde a um determinado padrão no conjunto corrente de regras. □

Definição 7 (linguagem reconhecida por um autômato adaptativo)**.** A linguagem reconhecida por um autômato adaptativo M é dada por $L(M) = \{w \in \Sigma^* \mid (E_0, q_0, w, Z_0) \vdash^* (E_n, f, \epsilon, Z_0), f \in F\}$. □

III. Aspectos técnicos e implementação

A ferramenta `XML2AA` permite a transformação automática de uma especificação XML de um autômato adaptativo em uma instância da classe `AdaptiveAutomaton` da biblioteca `AA4J` [5], pronta para uso direto em aplicações ou para consulta de cadeias através de um terminal integrado. Esta seção apresenta os aspectos técnicos e discussões sobre implementação.

A. Descrição formal da especificação XML

A ferramenta `XML2AA` requer uma descrição formal acerca do vocabulário e estrutura da especificação XML de um autômato adaptativo. Tal conjunto de regras é definido na Figura 1, em formato DTD (acrônimo de *Document Type Declaration*).

```
<!DOCTYPE adaptiveAutomaton [
   <!ELEMENT adaptiveAutomaton
        (transitions,submachines,actions?)>
   <!ELEMENT transitions (transition+)>
   <!ELEMENT transition
        (preAdaptiveFunction?,postAdaptiveFunction?)>
   <!ATTLIST transition to CDATA #REQUIRED>
   <!ATTLIST transition from CDATA #REQUIRED>
   <!ATTLIST transition symbol CDATA #IMPLIED>
   <!ATTLIST transition call CDATA #IMPLIED>
   <!ELEMENT parameter (#PCDATA)>
   <!ELEMENT submachines (submachine+)>
   <!ELEMENT submachine (state+)>
   <!ATTLIST submachine name CDATA #REQUIRED>
   <!ATTLIST submachine main CDATA #FIXED "true">
   <!ELEMENT state EMPTY>
   <!ATTLIST state name CDATA #REQUIRED>
   <!ATTLIST state start CDATA #FIXED "true">
   <!ATTLIST state accepting CDATA #FIXED "true">
   <!ELEMENT actions (adaptiveAction+)>
   <!ELEMENT adaptiveAction
        (parameter*,variable*,generator*,action+)>
   <!ATTLIST adaptiveAction name CDATA #REQUIRED>
   <!ELEMENT variable (#PCDATA)>
   <!ELEMENT generator (#PCDATA)>
   <!ELEMENT action
        (preAdaptiveFunction?,postAdaptiveFunction?)>
   <!ATTLIST action type (add|remove|query) #REQUIRED>
   <!ATTLIST action to CDATA #REQUIRED>
   <!ATTLIST action from CDATA #REQUIRED>
   <!ATTLIST action symbol CDATA #IMPLIED>
   <!ATTLIST action call CDATA #IMPLIED>
   <!ELEMENT preAdaptiveFunction (parameter*)>
   <!ATTLIST preAdaptiveFunction name CDATA #REQUIRED>
   <!ELEMENT postAdaptiveFunction (parameter*)>
   <!ATTLIST postAdaptiveFunction name CDATA #REQUIRED>
]>
```

Figura 1. Vocabulário e estrutura da especificação XML de um autômato adaptativo, em formato DTD.

De acordo com a Figura 1, a especificação XML de um autômato adaptativo é considerada bem formada se esta é aderente às regras gerais XML e válida se esta obedece às regras DTD que definem as etiquetas e atributos correspondentes, bem como a estrutura sintática.

B. Etiqueta principal

A especificação XML de um autômato adaptativo deve iniciar-se com a etiqueta principal `<adaptiveAutomaton>`, que denota a descrição dos componentes do dispositivo propriamente dito. Respectivamente, a etiqueta `</adaptiveAutomaton>` conclui a especificação.

De acordo com a Figura 1, a especificação do autômato deve conter, ao menos, um conjunto de transições e um conjunto de submáquinas. O terceiro componente é opcional, descrevendo as ações adaptativas presentes no modelo. A Figura 2 apresenta uma estrutura geral para a especificação XML de um autômato adaptativo.

```
<adaptiveAutomaton>
  <transitions> ... </transitions>
  <submachines> ... </submachines>
  <actions> ... </actions>
</adaptiveAutomaton>
```

Figura 2. Estrutura geral para a especificação XML de um autômato adaptativo. Os componentes `<transitions>` e `<submachines>` são obrigatórios.

É importante destacar que tais componentes são suficientemente expressivos para representar reconhecedores de linguagens regulares, livres de contexto e dependentes de contexto.

C. Transições

Uma transição do autômato adaptativo é representada através da etiqueta `<transition>` dentro da etiqueta ascendente `<transitions>`. Esta possui atributos que denotam qual o tipo de transição representado. Eventualmente, a etiqueta é identificada como vazia (isto é, não possui elementos internos), a menos da presença de funções adaptativas associadas. A Tabela I apresenta os atributos da etiqueta `<transition>` e seus significados correspondentes.

Tabela I
ATRIBUTOS DA ETIQUETA `<transition>` E SEUS SIGNIFICADOS CORRESPONDENTES.

Atributo	Significado
`from`	estado de origem
`to`	estado de destino
`symbol`	símbolo a ser consumido
`call`	chamada de submáquina

Conforme ilustra a Tabela I, os atributos `from` e `to` representam os estados de origem e destino, respectivamente, e são elementos obrigatórios em toda transição do modelo. Transições em vazio omitem o atributo `symbol`. Não é possível incluir atributos de consumo de símbolo e chamada de submáquina em uma mesma transição. Os estados são representados por identificadores inteiros positivos.

Exemplo 1. Considere o trecho de autômato apresentado na Figura 3, contendo transições com consumo de símbolos, em vazio e chamada de submáquina. A Figura 4 ilustra a especificação XML correspondente.

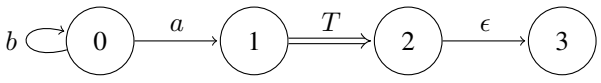

Figura 3. Trecho de autômato contendo transições com consumo de símbolos, em vazio e chamada de submáquina.

Observe que a etiqueta `<transition>` foi especificada em sua forma vazia. Entretanto, a forma longa da etiqueta é igualmente válida (e obrigatória quando existem funções adaptativas associadas). □

Uma transição pode conter funções adaptativas associadas. Para tal, utilizam-se as etiquetas internas `<preAdaptiveFunction>` e `<postAdaptiveFunction>`, denotando

```
<transitions>
  <transition from="0" symbol="a" to="1" />
  <transition from="0" symbol="b" to="0" />
  <transition from="1" call="T" to="2" />
  <transition from="2" to="3" />
</transitions>
```

Figura 4. Especificação XML das transições com consumo de símbolos, em vazio e chamada de submáquina do trecho de autômato da Figura 3.

as funções adaptativas anterior e posterior, respectivamente. Tais etiquetas requerem a presença do atributo `name` contendo o nome da função adaptativa a ser disparada. Eventualmente, as funções adaptativas possuem parâmetros associados, especificados através da sucessão de etiquetas `<parameter>` no corpo da etiqueta ascendente.

Exemplo 2. Considere o trecho de autômato apresentado na Figura 5, contendo transições com funções adaptativas associadas. A Figura 6 apresenta a especificação XML correspondente.

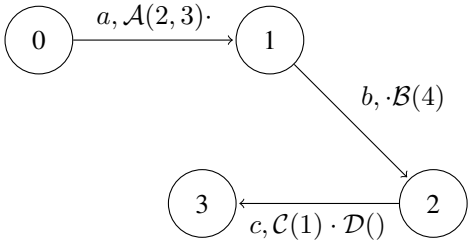

Figura 5. Trecho de autômato contendo transições com funções adaptativas associadas.

```
<transitions>

  <transition from="0" symbol="a" to="1">
    <preAdaptiveFunction name="A">
      <parameter>2</parameter>
      <parameter>3</parameter>
    </preAdaptiveFunction>
  </transition>

  <transition from="1" symbol="b" to="2">
    <postAdaptiveFunction name="B">
      <parameter>4</parameter>
    </postAdaptiveFunction>
  </transition>

  <transition from="2" symbol="c" to="3">
    <preAdaptiveFunction name="C">
      <parameter>1</parameter>
    </preAdaptiveFunction>
    <postAdaptiveFunction name="D" />
  </transition>

</transitions>
```

Figura 6. Especificação XML das transições com funções adaptativas associadas do trecho de autômato da Figura 5.

Observe que as etiquetas referentes às funções adaptativas

podem admitir formas vazias quando não existem parâmetros associados, isto é, nas chamadas de funções adaptativas que não sejam paramétricas. Cada transição admite, no máximo, duas funções adaptativas associadas. □

D. Submáquinas

Uma submáquina do autômato adaptativo é representada através da etiqueta `<submachine>` dentro da etiqueta ascendente `<submachines>`. Esta requer a presença do atributo `name` contendo o nome associado a tal submáquina, utilizado como identificador para eventuais chamadas, e um conjunto de estados correspondentes. O modelo do autômato requer que, ao menos, uma submáquina seja definida.

A especificação da submáquina pode conter ainda um atributo opcional `main`, indicando que esta é a submáquina principal do autômato adaptativo. Tal atributo, se presente, deve possuir o valor esperado `true`. Apenas uma submáquina pode conter o atributo `main`. A Tabela II ilustra os atributos da etiqueta `<submachine>`, seus significados correspondentes e valores esperados, quando aplicáveis.

Tabela II
ATRIBUTOS DA ETIQUETA `<submachine>`, SEUS SIGNIFICADOS CORRESPONDENTES E VALORES ESPERADOS, QUANDO APLICÁVEIS.

Atributo	Significado	Valor esperado
`name`	nome da submáquina	—
`main`	submáquina principal	`true`

Um estado é representado através da etiqueta vazia `<state>` e possui atributos que fornecem informações adicionais no contexto da submáquina na qual este está inserido. A Tabela III apresenta os atributos da etiqueta `<state>`, seus significados correspondentes e valores esperados, quando aplicáveis.

Tabela III
ATRIBUTOS DA ETIQUETA `<state>`, SEUS SIGNIFICADOS CORRESPONDENTES E VALORES ESPERADOS, QUANDO APLICÁVEIS.

Atributo	Significado	Valor esperado
`name`	nome do estado	—
`start`	estado inicial	`true`
`accepting`	estado de aceitação	`true`

De acordo com a Tabela III, um estado deve possuir um nome associado e, opcionalmente, conter indicações se este é o estado inicial ou um dos estados de aceitação da submáquina a qual está inserido. Os nomes dos estados são representados por identificadores inteiros positivos.

Exemplo 3. Considere as duas submáquinas T e U apresentadas na Figura 7, com T definida como submáquina principal. A Figura 8 apresenta a especificação XML correspondente.

Observe que, por uma questão de organização do modelo, os conjuntos de estados Q_T e Q_U das submáquinas T e U, respectivamente, são disjuntos, isto é, $Q_T \cap Q_U = \emptyset$. □

E. Ações

As ações adaptativas do autômato adaptativo são representadas através um conjunto de etiquetas `<adaptiveAction>` dentro

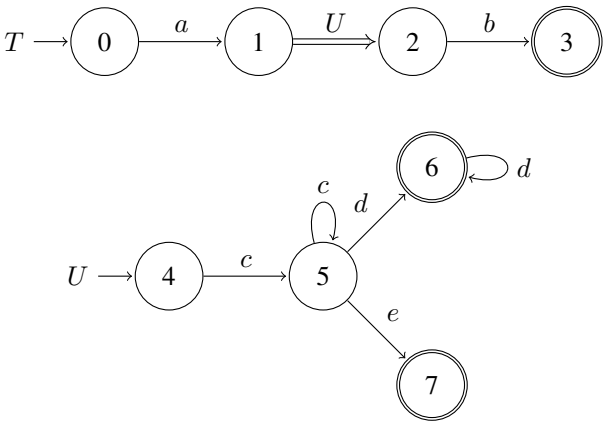

Figura 7. Submáquinas T e U. A submáquina T é definida como principal.

```
<submachines>

 <submachine name="T" main="true">
  <state name="0" start="true" />
  <state name="1" />
  <state name="2" />
  <state name="3" accepting="true" />
 </submachine>

 <submachine name="U">
  <state name="4" start="true" />
  <state name="5" />
  <state name="6" accepting="true" />
  <state name="7" accepting="true" />
 </submachine>

</submachines>
```

Figura 8. Especificação XML das submáquinas T e U da Figura 7. A submáquina T é definida como principal.

da etiqueta ascendente `<actions>`. Uma ação adaptativa requer a presença do atributo `name` contendo o nome associado a tal ação, utilizado como identificador para eventuais chamadas de funções adaptativas nas transições, e uma sequência não-vazia de ações adaptativas elementares.

Uma ação adaptativa elementar é representada através da etiqueta `<action>` e possui os mesmos atributos (Tabela I) e etiquetas descendentes associados a uma transição. Adicionalmente, uma ação adaptativa elementar requer o atributo `type`, que define o tipo de ação a ser aplicada, conforme ilustra a Tabela IV.

Tabela IV
VALORES PERMITIDOS PARA O ATRIBUTO `type` DA ETIQUETA `<action>`, REPRESENTANDO UMA AÇÃO ADAPTATIVA ELEMENTAR.

Valor	Significado
`query`	Realiza a inspeção do conjunto de regras
`remove`	Remove regras de acordo com um padrão
`add`	Insere regras que correspondem a um padrão

Exemplo 4. Considere o trecho de autômato da Figura 9, no qual uma função adaptativa $\mathcal{A}$ (Algoritmo 1) realiza alterações em sua topologia, removendo a transição $(1, b) \rightarrow 0$ e

inserindo uma nova transição $(0, a) \to 1$. A Figura 10 ilustra a especificação XML correspondente.

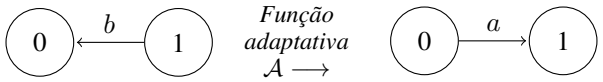

Figura 9. Trecho de autômato que ilustra uma função adaptativa removendo a transição $(1, b) \to 0$ e inserindo uma nova transição $(0, a) \to 1$.

Algoritmo 1 Função adaptativa $\mathcal{A}$

função adaptativa $\mathcal{A}$
$-(1, b) \to 0$
$+(0, a) \to 1$
fim da função adaptativa

```
<actions>

 <adaptiveAction name="A">
  <action type="remove" from="1" symbol="b" to="0" />
  <action type="add" from="0" symbol="a" to="1" />
 </adaptiveAction>

</actions>
```

Figura 10. Especificação XML da função adaptativa $\mathcal{A}$ do Algoritmo 1.

É importante observar que a especificação XML não admite uma sequência vazia de ações adaptativas elementares. □

Eventualmente, uma ação adaptativa pode conter parâmetros, variáveis e geradores. Tais elementos são disponibilizados de acordo com a especificação apresentada na Tabela V. Observe que as etiquetas não possuem atributos associados.

Tabela V
PARÂMETROS, VARIÁVEIS E GERADORES, DEFINIDOS COM SUAS ETIQUETAS E REGRAS DE FORMAÇÃO CORRESPONDENTES.

Etiqueta	Significado	Regra de formação
`<parameter>`	Parâmetro	Forma livre (não há restrição)
`<variable>`	Variável	Símbolo deve iniciar com `?`
`<generator>`	Gerador	Símbolo deve terminar com `*`

É importante observar que, ainda que não haja restrição quanto ao nome de um parâmetro na definição de uma ação adaptativa, é interessante evitar nomes que possam causar colisões potenciais com o alfabeto da linguagem (por exemplo, a colisão entre um símbolo `a` e um parâmetro `a` resultará na preferência pelo parâmetro, que possui maior prioridade).

Exemplo 5. Considere o trecho de autômato da Figura 11, no qual uma função adaptativa paramétrica $\mathcal{B}$ (Algoritmo 2) realiza alterações em sua topologia, invertendo os estados de origem e destino passados como parâmetros à função. A Figura 12 ilustra a especificação XML correspondente.

É importante destacar que as variáveis necessitam de preenchimento através de ações adaptativas elementares de inspeção. A utilização de uma variável não preenchida por uma ação adaptativa elementar de remoção ou inclusão resultará em uma exceção na biblioteca `AA4J` subjacente. □

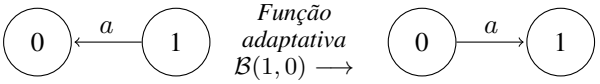

Figura 11. Trecho de autômato que ilustra uma função adaptativa invertendo os estados de origem e destino passados como parâmetros à função.

Algoritmo 2 Função adaptativa $\mathcal{B}(p_1, p_2)$

função adaptativa $\mathcal{B}(p_1, p_2)$
variáveis: $?x$
$?(p_1, ?x) \to p_2$
$-(p_1, ?x) \to p_2$
$+(p_2, ?x) \to p_1$
fim da função adaptativa

F. Modos de execução

A ferramenta `XML2AA` possui dois modos de operação. O primeiro é chamado *modo interativo*, no qual um arquivo XML é fornecido à ferramenta e um terminal de consulta para submissão de cadeias é disponibilizado ao usuário. A Figura 13 ilustra a organização do modo interativo. Observe que a biblioteca `AA4J` é uma dependência de execução.

De acordo com a Figura 13, a ferramenta `XML2AA` realiza o carregamento do arquivo XML e aplica transformações e validações na estrutura. Como resultado, uma instância do autômato adaptativo correspondente é gerada (utilizando a classe `AdaptiveAutomaton` da biblioteca `AA4J`) e um terminal de consulta para submissão de cadeias é disponibilizado ao usuário. A Figura 14 apresenta um exemplo de execução da ferramenta em modo interativo.

O terminal de consulta permite que o usuário submeta uma cadeia de símbolos ao autômato, através da ação `:check`, conforme ilustra a Figura 14. Como resultado, a ferramenta informa se a cadeia pertence à linguagem reconhecida pelo autômato, o tempo do processo de reconhecimento e o tipo de execução (determinística ou não-determinística). Adicionalmente, é possível obter uma representação gráfica da topologia do autômato, através da ação `:view` seguida de um índice positivo referente à topologia a ser visualizada. A ação `:quit` é utilizada para encerrar o terminal de consulta e o modo interativo da ferramenta.

O segundo modo de execução disponível é chamado *modo de biblioteca*, no qual a ferramenta é inserida no contexto

```
<actions>

 <adaptiveAction name="B">
  <parameter>p1</parameter>
  <parameter>p2</parameter>
  <variable>?x</variable>
  <action type="query" from="p1" symbol="?x" to="p2" />
  <action type="remove" from="p1" symbol="?x" to="p2" />
  <action type="add" from="p2" symbol="?x" to="p1" />
 </adaptiveAction>

</actions>
```

Figura 12. Especificação XML da função adaptativa $\mathcal{B}$ do Algoritmo 2.

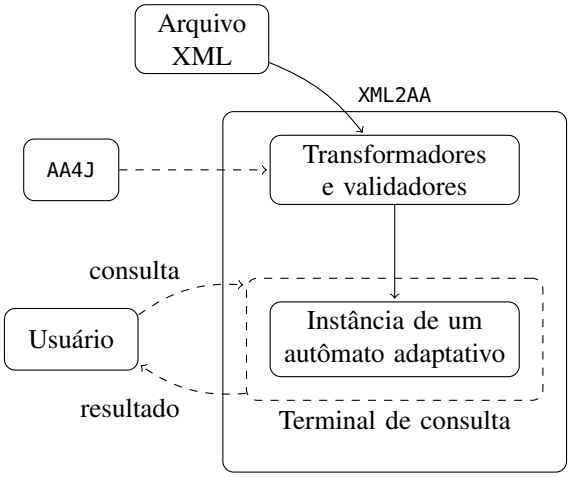

Figura 13. Organização da ferramenta XML2AA em modo interativo.

```
$ java -jar xml2aa.jar ape.xml
           _ ___
__ ___ __ | |_  )__ _ __ _
\ \ / '  \| |/ // _` / _` |
/_\_\_|_|_|_/___\__,_\__,_|

Laboratório de Linguagens e Técnicas Adaptativas
Escola Politécnica, Universidade de São Paulo
(versão 1.1)

O autômato possui 4 estados, 3 transições (2
transições com consumo de símbolo, 0 transições em
vazio, e 1 chamada de submáquina), 1 submáquina,
0 chamadas de funções adaptativas (0 funções
adaptativas anteriores, e 0 funções adaptativas
posteriores), e 0 ações adaptativas definidas.

Iniciando terminal, por favor, aguarde...
(pressione CTRL+C or digite ':quit' para sair do
    programa)

[1] consulta > :check ab
[1] resultado > cadeia aceita em 0:00:00.802
    (determinístico)

[2] consulta > :check aabb
[2] resultado > cadeia aceita em 0:00:00.024
    (determinístico)

[3] consulta > :quit

Isso é tudo, pessoal!
```

Figura 14. Exemplo de execução da ferramenta XML2AA em modo interativo.

de uma aplicação e atua como biblioteca propriamente dita, disponibilizando suas classes utilitárias para geração de uma instância de um autômato adaptativo (utilizando a classe `AdaptiveAutomaton` da biblioteca `AA4J`) a partir um arquivo XML contendo a especificação. A Figura 15 ilustra a organização do modo de biblioteca.

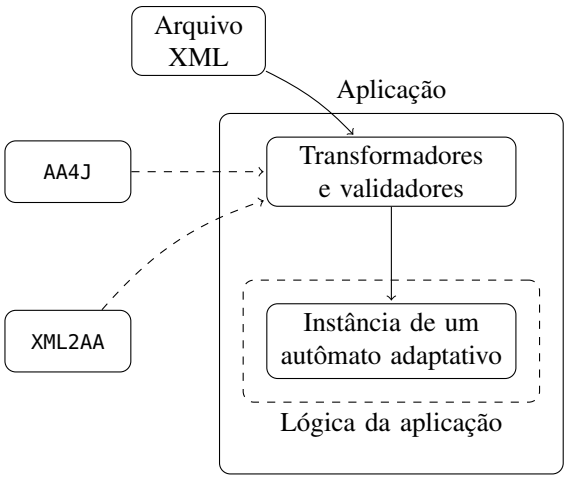

Figura 15. Organização da ferramenta XML2AA em modo de biblioteca.

De acordo com a Figura 15, a ferramenta XML2AA atua como biblioteca no contexto de uma aplicação e simplifica o processo de especificação de um autômato adaptativo diretamente em código Java (conforme exige a biblioteca AA4J), tornando-o transparente ao usuário, o qual deve apenas fornecer a especificação XML e tratar do modelo gerado em alto nível. A Figura 16 apresenta um exemplo da utilização da ferramenta em modo de biblioteca.

```
public class Main {
    public static void main(String[] args) {
        File file = new File("ape.xml");

        XMLTransformation transformation =
                                new XMLTransformation();
        XMLAdaptiveAutomaton xml = transformation.get(file);

        AutomatonValidator validator =
                                new AutomatonValidator(xml);
        validator.validate();

        AutomatonBuilder builder = new AutomatonBuilder();
        AdaptiveAutomaton automaton = builder.build(xml);
        ...
    }
}
```

Figura 16. Exemplo de utilização da ferramenta XML2AA em modo de biblioteca. Para fins didáticos, partes não essenciais do código Java foram omitidas.

A utilização da ferramenta em modo de biblioteca consiste, em linhas gerais, na transformação do arquivo XML em uma representação intermediária, validação dessa representação e construção efetiva da instância do autômato adaptativo correspondente. A biblioteca AA4J é uma dependência da ferramenta XML2AA e deve ser incluída no projeto da aplicação.

IV. Exemplos

Esta seção apresenta três exemplos de autômatos para ilustrar o funcionamento da ferramenta `XML2AA`. Todas as especificações XML correspondentes foram executadas em modo interativo.

Exemplo 6. Considere um autômato finito M_1 que reconhece cadeias pertencentes à linguagem regular $L_1 = \{w \in \{a,b\}^* \mid w = (ab)^+\}$, de acordo com a Figura 17. A especificação XML de tal autômato e a submissão das cadeias ab, $abab$ e aba são apresentadas nas Figuras 18 e 19, respectivamente. A janela de visualização da topologia do autômato finito M_1, obtida através da execução da ação `:view` da ferramenta, é apresentada na Figura 20.

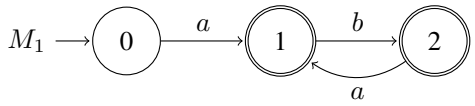

Figura 17. Autômato finito M_1 que reconhece cadeias pertencentes à linguagem regular $L = \{w \in \{a,b\}^* \mid w = (ab)^+\}$.

```
<adaptiveAutomaton>

 <transitions>
  <transition from="0" symbol="a" to="1" />
  <transition from="1" symbol="b" to="2" />
  <transition from="2" symbol="a" to="1" />
 </transitions>

 <submachines>
  <submachine name="M1" main="true">
   <state name="0" start="true" />
   <state name="1" />
   <state name="2" accepting="true"/>
  </submachine>
 </submachines>

</adaptiveAutomaton>
```

Figura 18. Especificação XML do autômato finito M_1 da Figura 17.

De acordo com a submissão de cadeias ilustrada na Figura 19, $ab, abab \in L(M_1)$ e $aba \notin L(M_1)$. □

Exemplo 7. Considere um autômato de pilha estruturado M_2 que reconhece cadeias pertencentes à linguagem livre de contexto $L_2 = \{w \in \{a,b\}^* \mid w = a^n b^n, n \in \mathbb{Z}\}$, de acordo com a Figura 21. A especificação XML de tal autômato e a submissão das cadeias ab, aab e $aabb$ são apresentadas nas Figuras 22 e 23, respectivamente. A janela de visualização da topologia do autômato de pilha estruturado M_2, obtida através da execução da ação `:view` da ferramenta, é apresentada na Figura 24.

De acordo com a submissão de cadeias ilustrada na Figura 23, $ab, aabb \in L(M_2)$ e $aab \notin L(M_2)$. □

Exemplo 8. Considere um autômato adaptativo M_3 que reconhece cadeias pertencentes à linguagem dependente de contexto $L_3 = \{w \in \{a,b,c\}^* \mid w = a^n b^n c^n, n \in \mathbb{Z}, n \geq 1\}$, de acordo com a Figura 25. A função adaptativa $\mathcal{A}$ é apresentada

```
[1] consulta > :check ab
[1] resultado > cadeia aceita em 0:00:00.461
    (determinístico)

[2] consulta > :check abab
[2] resultado > cadeia aceita em 0:00:00.013
    (determinístico)

[3] consulta > :check aba
[3] resultado > cadeia rejeitada em 0:00:00.011
    (determinístico)

[4] consulta > :view 1
[4] resultado > autômato visualizado em janela externa.
```

Figura 19. Submissão das cadeias ab, $abab$ e aba à instância do autômato finito M_1 gerado a partir da especificação XML da Figura 18.

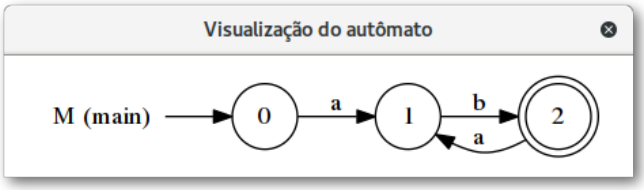

Figura 20. Visualização da topologia do autômato finito M_1.

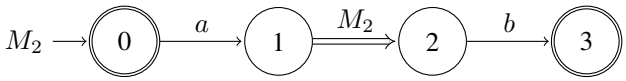

Figura 21. Autômato de pilha estruturado M_2 que reconhece cadeias pertencentes à linguagem livre de contexto $L_2 = \{w \in \{a,b\}^* \mid w = a^n b^n, n \in \mathbb{Z}\}$.

```
<adaptiveAutomaton>

 <transitions>
  <transition from="1" symbol="a" to="2" />
  <transition from="2" call="M2" to="3" />
  <transition from="3" symbol="b" to="4" />
 </transitions>

 <submachines>
  <submachine name="M2" main="true">
   <state name="1" start="true" accepting="true" />
   <state name="2" />
   <state name="3" />
   <state name="4" accepting="true"/>
  </submachine>
 </submachines>

</adaptiveAutomaton>
```

Figura 22. Especificação XML do autômato de pilha estruturado M_2 da Figura 21.

```
[1] consulta > :check ab
[1] resultado > cadeia aceita em 0:00:00.503
    (determinístico)

[2] consulta > :check aab
[2] resultado > cadeia rejeitada em 0:00:00.018
    (determinístico)

[3] consulta > :check aabb
[3] resultado > cadeia aceita em 0:00:00.016
    (determinístico)

[4] consulta > :view 1
[4] resultado > autômato visualizado em janela externa.
```

Figura 23. Submissão das cadeias ab, aab e $aabb$ à instância do autômato de pilha estruturado M_2 gerado a partir da especificação XML da Figura 22.

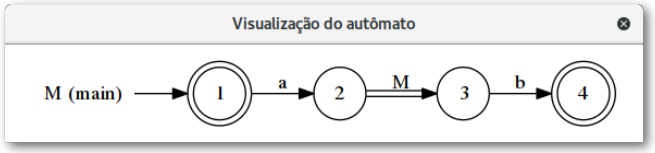

Figura 24. Visualização da topologia do autômato de pilha estruturado M_2.

no Algoritmo 3. A especificação de tal autômato e a submissão das cadeias $aabbc$, $aabbcc$ e $aaabbbccc$ são apresentadas nas Figuras 26 e 27, respectivamente. As janelas de visualização das topologias do autômato adaptativo M_3 referentes aos reconhecimentos das cadeias $aabbcc$ e $aaabbbccc$ são apresentadas nas Figuras 28 e 29, respectivamente.

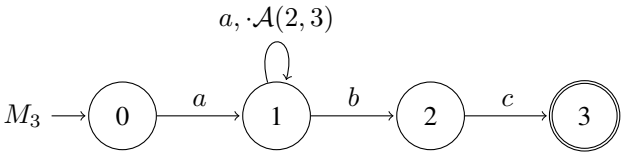

Figura 25. Autômato adaptativo M_3 que reconhece cadeias pertencentes à linguagem dependente de contexto $L_3 = \{w \in \{a,b,c\}^* \mid w = a^n b^n c^n, n \in \mathbb{Z}, n \geq 1\}$.

De acordo com a submissão de cadeias ilustrada na Figura 27, $aabbcc, aaabbbccc \in L(M_3)$ e $aabbc \notin L(M_3)$. □

V. Considerações finais

O código-fonte da ferramenta `XML2AA` está disponível para consulta e download em `https://github.com/cereda/xml2aa`, sob a licença GPLv3[1]. O bytecode gerado é compatível com Java 8 ou versões superiores. Espera-se que esta contribuição possa proporcionar subsídios para a implementação de programas que apresentem características adaptativas, através de uma especificação simplificada utilizando a notação XML, atuando como uma interface amigável à biblioteca `AA4J`.

[1]Disponível em `http://www.gnu.org/licenses/gpl-3.0.html`.

```
<adaptiveAutomaton>

 <transitions>
  <transition from="0" symbol="a" to="1"/>
  <transition from="1" symbol="b" to="2"/>
  <transition from="2" symbol="c" to="3"/>
  <transition from="1" symbol="a" to="1" >
   <postAdaptiveFunction name="A">
    <parameter>2</parameter>
    <parameter>3</parameter>
   </postAdaptiveFunction>
  </transition>
 </transitions>

 <submachines>
  <submachine name="M" main="true">
   <state name="0" start="true" />
   <state name="1" />
   <state name="2" />
   <state name="3" accepting="true"/>
  </submachine>
 </submachines>

 <actions>
  <adaptiveAction name="A">
   <parameter>p1</parameter>
   <parameter>p2</parameter>
   <variable>?x</variable>
   <variable>?y</variable>
   <generator>g1*</generator>
   <generator>g2*</generator>
   <action type="query" from="?x" symbol="b" to="p1"/>
   <action type="remove" from="?x" symbol="b" to="p1"/>
   <action type="query" from="?y" symbol="c" to="p2"/>
   <action type="remove" from="?y" symbol="c" to="p2"/>
   <action type="remove" from="1" symbol="a" to="1">
    <postAdaptiveFunction name="A">
     <parameter>p1</parameter>
     <parameter>p2</parameter>
    </postAdaptiveFunction>
   </action>
   <action type="add" from="?x" symbol="b" to="g1*"/>
   <action type="add" from="g1*" symbol="b" to="p1"/>
   <action type="add" from="?y" symbol="c" to="g2*"/>
   <action type="add" from="g2*" symbol="c" to="p2"/>
   <action type="add" from="1" symbol="a" to="1">
    <postAdaptiveFunction name="A">
     <parameter>g1*</parameter>
     <parameter>g2*</parameter>
    </postAdaptiveFunction>
   </action>
  </adaptiveAction>
 </actions>

</adaptiveAutomaton>
```

Figura 26. Especificação XML do autômato adaptativo M_3 da Figura 25.

Algoritmo 3 Função adaptativa $\mathcal{A}(p_1, p_2)$

função adaptativa $\mathcal{A}(p_1, p_2)$
variáveis: $?x, ?y$
geradores: $g_1^\star, g_2^\star$
$?(?x, b) \rightarrow p_1$
$-(?x, b) \rightarrow p_1$
$?(?y, c) \rightarrow p_2$
$-(?y, c) \rightarrow p_2$
$-(1, a) \rightarrow 1, \cdot\mathcal{A}(p_1, p_2)$
$+(?x, b) \rightarrow g_1^\star$
$+(g_1^\star, b) \rightarrow p_1$
$+(?y, c) \rightarrow g_2^\star$
$+(g_2^\star, c) \rightarrow p_2$
$+(1, a) \rightarrow 1, \cdot\mathcal{A}(g_1^\star, g_2^\star)$
fim da função adaptativa

```
[1] consulta > :check aabbc
[1] resultado > cadeia rejeitada em 0:00:00.463
    (determinístico)

[2] consulta > :check aabbcc
[2] resultado > cadeia aceita em 0:00:00.014
    (determinístico)

[3] consulta > :view 1
[3] resultado > autômato visualizado em janela externa.

[4] consulta > :view 2
[4] resultado > autômato visualizado em janela externa.

[5] consulta > :check aaabbbccc
[5] resultado > cadeia aceita em 0:00:00.030
    (determinístico)

[6] consulta > :view 1
[6] resultado > autômato visualizado em janela externa.

[7] consulta > :view 2
[7] resultado > autômato visualizado em janela externa.

[8] consulta > :view 3
[8] resultado > autômato visualizado em janela externa.
```

Figura 27. Submissão das cadeias $aabbc$, $aabbcc$ e $aaabbbccc$ à instância do autômato adaptativo M_3 gerado a partir da especificação XML da Figura 26.

A ferramenta apresentada neste artigo pode atuar como material didático no ensino de autômatos adaptativos, através de seu modo interativo, permitindo uma especificação em alto nível (utilizando a notação XML) e posterior submissão de cadeias no terminal de consulta. Adicionalmente, a ferramenta pode contribuir com a implementação de autômatos adaptativos utilizando a linguagem Java de forma consistente e aderente à teoria, através do modo de biblioteca.

AGRADECIMENTOS

Os autores agradecem ao professor Ítalo Santiago Vega, do Departamento de Ciência da Computação da Pontifícia Universidade Católica de São Paulo, pelas valiosas contribuições para o desenvolvimento da ferramenta e escrita do artigo.

REFERÊNCIAS

[1] J. José Neto, "Contribuições à metodologia de construção de compiladores," Tese de livre docência, Escola Politécnica da Universidade de São Paulo, São Paulo, 1993.

[2] J. José Neto and M. E. S. Magalhães, "Reconhecedores sintáticos: Uma alternativa didática para uso em cursos de engenharia," in *XIV Congresso Nacional de Informática*, 1981, pp. 171–181.

[3] J. José Neto, "Adaptive automata for context-sensitive languages," *SIGPLAN Notices*, vol. 29, no. 9, pp. 115–124, set 1994.

[4] ——, "Adaptive rule-driven devices: general formulation and case study," in *International Conference on Implementation and Application of Automata*, 2001.

[5] P. R. M. Cereda and J. José Neto, "AA4J: uma biblioteca para implementação de autômatos adaptativos," in *Memórias do X Workshop de Tecnologia Adaptativa – WTA 2016*, 2016, pp. 16–26.

Paulo Roberto Massa Cereda é graduado em Ciência da Computação pelo Centro Universitário Central Paulista (2005) e mestre em Ciência da Computação pela Universidade Federal de São Carlos (2008). Atualmente, é doutorando do Programa de Pós-Graduação em Engenharia de Computação do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo, atuando como aluno pesquisador no Laboratório de Linguagens e Técnicas Adaptativas do PCS. Tem experiência na área de Ciência da Computação, com ênfase em Teoria da Computação, atuando principalmente nos seguintes temas: tecnologia adaptativa, autômatos adaptativos, dispositivos adaptativos, linguagens de programação e construção de compiladores.

João José Neto é graduado em Engenharia de Eletricidade (1971), mestre em Engenharia Elétrica (1975), doutor em Engenharia Elétrica (1980) e livre-docente (1993) pela Escola Politécnica da Universidade de São Paulo. Atualmente, é professor associado da Escola Politécnica da Universidade de São Paulo e coordena o LTA – Laboratório de Linguagens e Técnicas Adaptativas do PCS – Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP. Tem experiência na área de Ciência da Computação, com ênfase nos Fundamentos da Engenharia da Computação, atuando principalmente nos seguintes temas: dispositivos adaptativos, tecnologia adaptativa, autômatos adaptativos, e em suas aplicações à Engenharia de Computação, particularmente em sistemas de tomada de decisão adaptativa, análise e processamento de linguagens naturais, construção de compiladores, robótica, ensino assistido por computador, modelagem de sistemas inteligentes, processos de aprendizagem automática e inferências baseadas em tecnologia adaptativa.

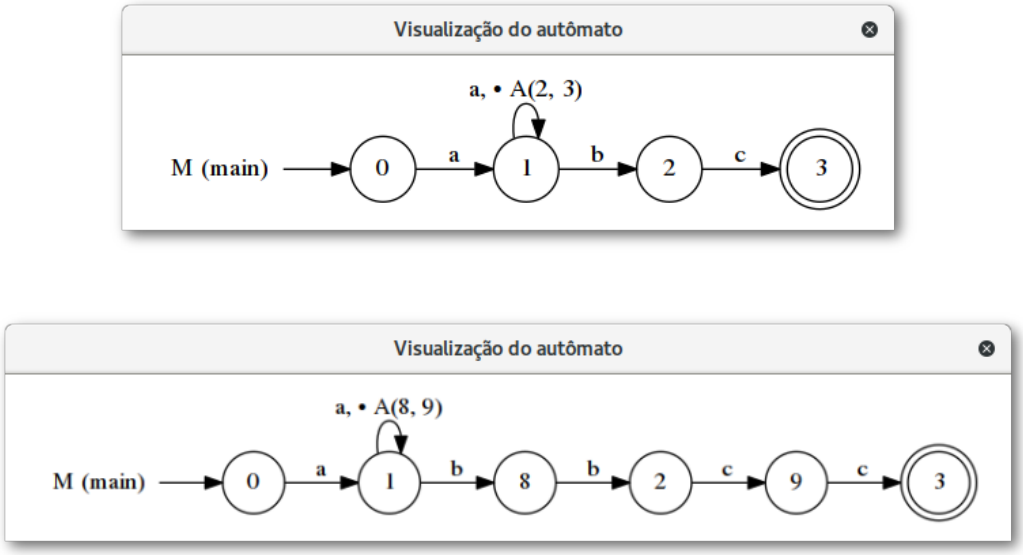

Figura 28. Visualização das topologias inicial e final do autômato adaptativo M_3, após o reconhecimento da cadeia $aabbcc$.

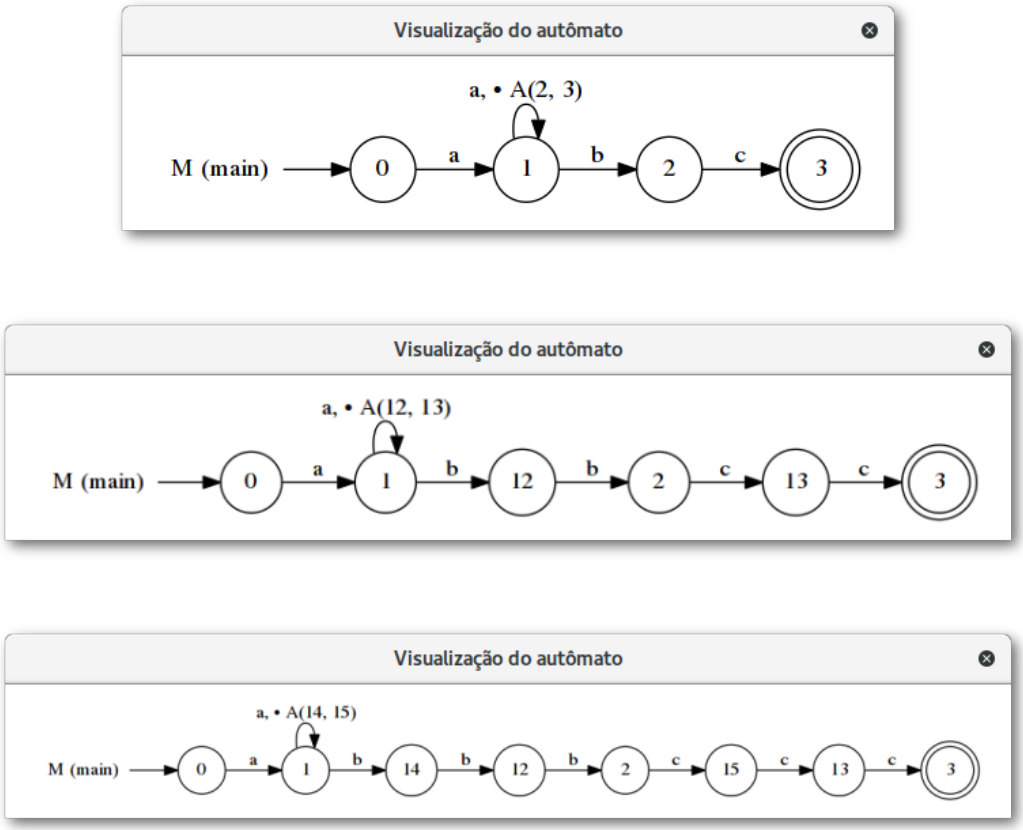

Figura 29. Visualização das topologias inicial, intermediária e final do autômato adaptativo M_3, após o reconhecimento da cadeia $aaabbbccc$.