TEMA Tend. Mat. Apl. Comput., **12**, No. 3 (2011), 171-182.
doi: 10.5540/tema.2011.012.03.0171

λ-ALN: Autômatos Lineares Não-Determinísticos com λ-Transições

B.C. BEDREGAL[1], Grupo de Lógica, Linguagem, Informação, Teoria e Aplicações – LoLITA, Departamento de Informática e Matemática Aplicada – DIMAp, Universidade Federal do Rio Grande do Norte – UFRN, Campus Universitário, 59072-970 Natal, Brazil.

Resumo. Neste trabalho introduziremos a classe dos autômatos lineares não-determinísticos com λ-transições. Baseados numa nova forma normal para gramáticas lineares, provamos que a classe de linguagens aceita por este tipo de autômato é exatamente a classe das linguagens lineares. Mostramos ainda que, análogo ao que ocorre com os autômatos finitos e com os autômatos com pilhas, a existência das λ-transições em um autômato linear não-determinístico não significa que não possa ser definido um autômato linear não-determinístico sem λ-transições que reconheça a mesma linguagem. Ou seja, as λ-transições não aumentam o poder de aceitação destes autômatos e portanto podem ser dispensadas do modelo. Finalmente, apresentamos uma aplicação destes autômatos no contexto de bioinformática.

Palavras-chave. Linguagens lineares, gramáticas lineares, forma normal, autômatos lineares não-determinísticos, λ-transições.

1. Introdução

A classe das linguagens lineares se localiza dentro da hierarquia de Chomsky entre as linguagens regulares e as livres de contexto. O principal modelo computacional para esta classe de linguagens são as gramáticas lineares. Esta gramática permite gerar cadeias fazendo casamentos entre prefixos e sufixos de subcadeias da cadeia a ser gerada. Esta capacidade de casamentos faz com que esta classe de linguagens contenha a maioria das linguagens livres de contexto usuais. Assim, por exemplo a linguagem dos palíndromos e $\{a^n b^n : n \geq 1\}$, que geralmente são usadas como exemplos de linguagens livres de contexto que não são regulares, são na verdade linguagens lineares. Um exemplo de uma linguagem que é livre de contexto e que não é linear é: $\{a^n b^n a^m b^m : n, m \geq 1\}$.

Em termos de autômatos, na literatura há pelo menos três modelos diferentes para a classe das linguagens lineares: Um tipo especial de autômato finito de duas fitas [22, 12], tradutores finitos [22, 20] e autômatos com pilha que executam no máximo uma volta para qualquer entrada [11, 12, 14, 3, 13]. Apesar dos méritos de cada um desses modelos, eles não são naturais nem intrínsecos, pois consideram

[1]bedregal@dimap.ufrn.br

Recebido em 08 Setembro 2010; Aceito em 01 Julho 2011. Apresentado no XXXIII CNMAC

elementos externos ao modelo. Por exemplo, no tipo especial de autômato finito de duas fitas, a entrada é dividida nas duas fitas, de tal modo que a segunda fita contém o reverso do pedaço mais à direita da cadeia de entrada. Note que os autômatos usuais (autômatos finitos determinísticos, autômatos com pilha, máquinas de Turing – mesmo a de duas fitas –, etc.) assumem que a entrada inteira está na fita de entrada ao momento de iniciar a computação. Assim, este modelo não possui esta característica natural. Além disso, a escolha de onde dividir assim como a reversão da parte direita é feita externa ao modelo. Portanto, este modelo não é nem natural nem intrínseco. O tradutor finito também divide a entrada em duas partes, mas as coloca numa única fita separadas por um símbolo especial, sendo que a segunda parte é invertida. Assim, analogamente ao modelo anterior, tradutores finitos não são intrínsecos nem naturais. Finalmente, uma vez que a volta no autômato com pilha é um movimento o qual decresce a pilha precedido por outro que aumenta a pilha, o autômato com pilha com no máximo uma volta é claramente não natural. Uma vez que a restrição ao autômato com pilha de poder fazer no máximo uma volta não faz parte da estrutura do autômato (só analisamos a posteriori se o autômato com pilha satisfaz ou não esta restrição) este modelo também não é intrínseco.

Neste artigo propomos um modelo de autômatos alternativos para a classe das linguagens lineares que chamaremos *autômatos lineares não-determinísticos com λ-transições*, que do nosso ponto de vista é mais natural e simples que os mostrados anteriormente. Além disso, nosso modelo é intrínseco e próprio, no sentido que não é uma restrição de algum tipo de autômato. Outra vantagem do autômato proposto com respeito aos outros modelos é que ele estende os autômatos finitos não-determinísticos (AFN), ou seja todo AFN é um autômato linear não-determinístico (ALN).

Por outro lado, os λ-movimentos nas classes habituais de autômatos não são essenciais para os modelos. Assim, por exemplo, λ-movimento não aumenta o poder computacional de aceitação de autômatos finitos (ver, por exemplo, [14]), nem de um autômato com pilha (ver, por exemplo, [12]) e nem das máquinas de Turing (Tese de Church). Neste trabalho, provamos que os ALN não têm seu poder de aceitação aumentado em função da utilização de λ-movimentos.

O artigo está organizado como segue. Na seção 2. apresentamos o conceito de gramáticas lineares e das linguagens geradas por tais gramáticas assim como duas novas formas normais para essas gramáticas. Na seção 3. introduzimos o conceito de autômatos lineares não-determinísticos com λ-transições e as linguagens aceitas por estes autômatos enquanto na seção seguinte mostramos que a classe das linguagens aceitas por esses autômatos é a classe das linguagens lineares. Na seção 5. mostramos que os λ-movimentos não aumentam o poder de aceitação dos autômatos lineares não-determinísticos e portanto podemos prescindir deles. Na seção 6. apresentamos uma aplicação dos ALN no estudo de sequências de DNA palindrómicas. Finalmente, a seção 7. apresenta algumas conclussões e perspectivas de trabalhos futuros.

2. Gramáticas Lineares

Como usual, uma gramática formal G é uma quádrupla $\langle V, T, S, P\rangle$ onde V é um conjunto finito de símbolos variáveis, T é um conjunto finito de símbolos terminais e portanto $V \cap T = \emptyset$, $S \in V$ é a variável de início e $P \subseteq (V \cup T)^+ \times (V \cup T)^*$ é o conjunto de produções. Pares ordenados $(x, y) \in P$ são denotados por $x \to y$.

Definição 2.1. *Uma gramática $G = \langle V, T, S, P\rangle$ é **linear**, se cada $x \to y \in P$ é tal que $x = A$ e $y = uBv$ para algum $A, B \in V$ e $u, v \in T^*$. Uma variável $A \in V$ será chamada **linear à esquerda** se cada produção $A \to y \in P$ ou $y \in T^*$ ou $y = Bz$ para algum $z \in T^+$ e $B \in V$. Analogamente, uma variável $A \in V$ será chamada **linear à direita** se cada produção $A \to y \in P$ ou $y \in T^*$ ou $y = zB$ para algum $z \in T^+$ e $B \in V$. Uma gramática linear G está na **forma normal linear** (FNL), se cada variável $A \in V$ ou é linear à direita ou é linear à esquerda.*

Assim, em uma gramática linear, cada produção em P tem no lado esquerdo uma variável e no lado direito uma cadeia com no máximo uma variável (o restante, se houver, são símbolos terminais), sem qualquer restrição da posição dessa eventual variável.

Note que uma variável A em uma gramática linear será linear à direita e linear à esquerda, ao mesmo tempo, apenas quando todas as produções tendo A no lado esquerdo, têm no seu lado direito uma cadeia composta apenas por símbolos terminais. Note também que diferentemente da FNL, a forma normal linear usual (FNLU) [23, 12, 5], permite que haja duas produções com o mesmo lado esquerdo, mas com seus lados direitos direitos diferindo na posição da variável: em uma das produções, a variável ocorre na posição mais à esquerda, enquanto na outra produção a variável ocorre na posição mais à direita. Observe que, se uma gramática linear está na FNL então também está na FNLU, porém uma gramática na FNLU pode não estar na FNL.

Como usual, para cada $u, v, w \in (V \cup T)^*$ e $A \in V$, $uAw \Rightarrow uvw$ se existe uma produção $A \to v \in P$. Seja $\Rightarrow^*$ o fecho reflexivo e transitivo de $\Rightarrow$. A **linguagem gerada** por uma gramática G é

$$L(G) = \{w \in T^* : S \Rightarrow^* w\}$$

Linguagens geradas por gramáticas lineares são chamadas **linguagens lineares**.

Lema 2.1. *Seja G uma gramática linear. Existe uma gramática linear G' na* FNL *tal que $L(G) = L(G')$.*

Demonstração. Sem perda de generalidade, podemos assumir que a gramática G não contém produções unitárias[2]. Para cada $A \in V$ e $u \in T^+$ seja $A_u = \{A \to uBv \in P$: para algum $B \in V$ e $v \in T^+\}$. Se $A_u \neq \emptyset$ então substitua cada produção $A \to uBv \in A_u$ em P pelas produções $A \to uC$ e $C \to Bv$, onde C é uma variável nova, e adicione C a V. Neste ponto, cada produção tem a forma:

[2]Produções unitárias são produções da forma $A \to B$. A transformação de uma gramática linear em uma gramática linear equivalente (ou seja que gera a mesma linguagem) livre de tais produções, pode ser feita aplicando o mesmo algoritmo usado para remoção de produções unitárias em gramáticas livres de contexto [14, 19, 5].

$$A \to uB, A \to Bu \text{ or } A \to u$$

para algum $u \in T^*$ e $A, B \in V$. Se a variável $A \in V$ não é nem linear à direita nem linear à esquerda, então substitua cada produção $A \to Bv \in P$ pelas produções $A \to C$ e $C \to Bv$, onde C é uma variável nova, e adicione C a V. A gramática resultante está na FNL e é equivalente à gramática original. □

Uma gramática linear G está na **forma normal linear forte** (FNLF) se está na FNL e os lados direitos das produções são da forma aA ou Aa para algum $a \in T \cup \{\lambda\}$ e $A \in V \cup \{\lambda\}$.

Proposição 2.1. *Seja G uma gramática linear. Existe uma gramática linear $\widehat{G}$ na* FNLF *tal que $L(G) = L(\widehat{G})$.*

Demonstração. Primeiro obtenha a FNL de G seguindo o algoritmo descrito no Lema 2.1 e em seguida aplique o seguinte algoritmo:

Para cada $A \in V$ e $a \in T$, se $A^a = \{A \to ya \in P \ : \ |y| \geq 2 \text{ ou } y \in T\} \neq \emptyset$, então adicione uma variável nova B a V, remova de P as produções em A^a e introduza as produções $A \to Ba$ e $B \to y$, para cada $A \to ya \in A^a$.

Depois, de forma análoga, para cada $A \in V$ e $a \in T$ se $A_a = \{A \to ay \in P \ : \ |y| \geq 2 \text{ ou } y \in T\} \neq \emptyset$, então adicione uma variável nova B a V, remova de P as produções em A_a e introduza as produções $A \to aB$ e $B \to y$, para cada $A \to ay \in A_a$.

O resultado é uma gramática linear na FNLF equivalente a G. □

3. Autômato Linear Não-Determinístico com λ-Transições

Um **autômato linear não-determinístico com λ-transições**[3] (λ-ALN), consiste de dois conjuntos finitos disjuntos de estados (Q e P) alguns dos quais serão considerados como estados de aceitação, uma fita de entrada infinita à direita e dividida em células, cada uma das quais comporta um símbolo de um alfabeto de entrada (ou o símbolol especial que representa o conteúdo "em branco"), duas cabeças leitoras e uma unidade de controle que administra a conduta do λ-ALN de acordo com a configuração corrente. A execução do λ-ALN começa com uma cadeia na fita de entrada, com a cabeça leitora esquerda apontando para o símbolo mais à esquerda na fita e a cabeça leitora direita apontando para o símbolo (diferente de branco) mais à direita na fita e tendo como estado corrente um estado pertence a um subconjunto especial de $Q \cup P$, chamado de conjunto de estados iniciais. Um passo computacional num λ-ALN é feito como segue: a unidade de controle, em função da classe que pertence o estado corrente, usa a cabeça leitora esquerda ou direita para ler um símbolo da fita, logo move uma célula para a direita a cabeça

[3] O termo linear aqui é devido ao fato de aceitar linguagens lineares diferentemente dos autômatos finitos lineares introduzidos em [24, p.14] onde o termo linear diz respeito à função de transição ser uma função linear.

leitora esquerda, caso o estado corrente pertença a Q, ou uma célula para esquerda a cabeça leitora direita, caso o estado corrente pertença a P. Ao mesmo tempo muda, seguindo uma escolha não-deterministica, para um estado de um conjunto de possíveis estados. A unidade de controle de um λ-ALN também admite mudar de estado sem mover as cabeças leitoras, isto é sem ler símbolos da fita de entrada. A computação pára quando uma cabeça leitora cruza a outra, caso o estado corrente nesse momento seja de aceitação o λ-ALN aceita a cadeia dada como entrada, caso contrário a rejeita. A figura 1 ilustra uma representação esquemática de um λ-ALN.

Formalmente, um λ-ALN é uma sêxtupla $M = \langle Q, P, \Sigma, \delta, I, F\rangle$ onde Q e P são conjuntos finitos disjuntos de estados, Σ é um conjunto finito de símbolos de entrada (alfabeto), $I \subseteq Q \cup P$ é o conjunto de estados de início, $F \subseteq (Q \cup P)$ é um conjunto de estados de aceitação e $\delta : (Q \cup P) \times (\Sigma \cup \{\lambda\}) \longrightarrow \wp(Q \cup P)$ é a função de transição[4].

Analogamente a autômatos finitos, cada λ-ALN tem associado um grafo dirigido chamado diagrama de transição. A única diferença com os diagramas de transição de autômatos finitos está na forma como distinguimos os estados de Q com os estados de P. Para os primeiros usamos círculos enquanto para os outros usamos quadrados. Assim, um autômato finito não-determinístico pode ser visto como um λ-ALN onde $P = \emptyset$. Portanto, λ-ALN é uma extensão natural para autômatos finitos λ-não-determinísticos.

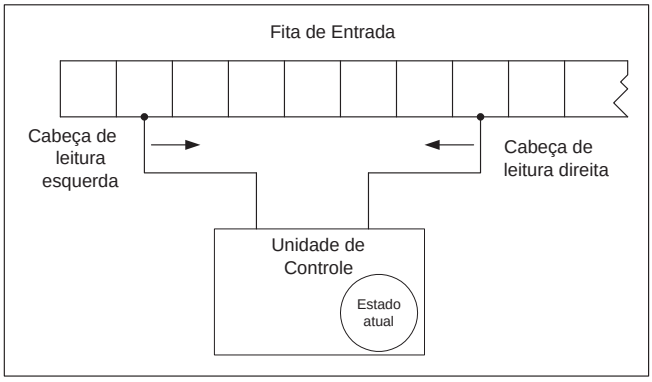

Figura 1: Representação esquemática de um λ-ALN.

Uma **descrição instantânea** (DI) de um λ-ALN deve registrar o estado corrente, o que resta ler da cadeia de entrada e qual cabeça está ativa. Assim, uma DI é um par $(q, w) \in (Q \cup P) \times \Sigma^*$ significando que resta ainda ler w na fita e que o estado atual é q. A cabeça leitora que está ativa é deduzida a partir do estado corrente.

O símbolo $\vdash$ denotará um movimento de uma DI a outra DI. Assim, para cada $q \in Q$, $q\prime \in Q \cup P$, $p \in P$, $w \in \Sigma^*$ e $a \in \Sigma \cup \{\lambda\}$

[4]Observe que o uso de um conjunto de estados como sendo estados iniciais não é artificial, uma vez que há diversos autores que usam conjuntos de estados iniciais em sua definição de autômatos. Veja por exemplo, [7, Def.4.1] e [25, Def.9] para autômatos finitos não-determinísticos e [12, pag.52] (ou [8, pag.89]) para o caso de sistemas (ou grafos) de transição.

$(q, aw) \vdash (q\prime, w)$ é possível se, e somente se, (sss) $q\prime \in \delta(q, a)$ e $(p, wa) \vdash (q\prime, w)$ é possível sss $q\prime \in \delta(p, a)$.

Usamos $\vdash^*$ para o fecho reflexivo e transitivo de $\vdash$, ou seja $\vdash^*$ representa uma quantidade arbitrária de movimentos.

A linguagem aceita por um λ-ALN M é o conjunto

$$L(M) = \{w \in \Sigma^* : (q_0, w) \vdash^* (q_f, \lambda) \text{ para algum } q_f \in F \text{ e } q_0 \in I\}$$

4. λ-ALN's e Linguagens Lineares

Primeiro mostraremos que cada linguagem aceita por algum λ-ALN é linear.

Teorema 4.1. *Seja $M = \langle Q, P, \Sigma, \delta, I, F\rangle$ um λ-ALN. Então existe uma gramática linear G tal que $L(M) = L(G)$.*

Demonstração. Seja $G = \langle Q \cup P \cup \{S\}, \Sigma, S, P'\rangle$ onde

$$\begin{aligned} P' = \quad & \{q_i \to aq_j : a \in \Sigma \cup \{\lambda\},\ q_i \in Q, q_j \in Q \cup P \text{ e } q_j \in \delta(q_i, a)\} \cup \\ & \{p_i \to q_j a : a \in \Sigma \cup \{\lambda\},\ p_i \in P, q_j \in Q \cup P \text{ e } q_j \in \delta(q_i, a)\} \cup \\ & \{q_f \to \lambda : q_f \in F\} \cup \{S \to q_0 : q_0 \in I\} \end{aligned}$$

Claramente, $L(M) = L(G)$. □

Observação 4.1. *Obviamente quando $I = \{q_0\}$ para algum $q_0 \in Q \cup P$, podemos usar o próprio q_0 como variável inicial e apagar toda ocorrência de S em G. Incluso no caso de $I \subseteq Q$ ou $I \subseteq P$ poder-se-ia usar a técnica de eliminação de produções unitárias usualmente aplicadas em gramáticas livres de contexto, conforme exemplificado em [14, 7, 19, 5].*

Note que a gramática G resultante da prova está na FNLF.

Agora provaremos que cada linguagem linear é aceita por um λ-ALN.

Teorema 4.2. *Seja $G = \langle V, T, S, \widehat{P}\rangle$ uma gramática linear. Então existe um λ-ALN M tal que $L(G) = L(M)$.*

Demonstração. Sem perda de generalidade podemos assumir que G está na FNLF. Seja $Q = \{A \in V : A \text{ é uma variável linear à direita}\} \cup \{C\}$, onde $C \notin V$ e $P = \{A \in V : A \text{ é uma variável linear à esquerda}\}$. Então,

$$M = \langle Q, P, T, \delta, \{S\}, F\rangle,$$

onde $F = \{C\}$ e para cada $A \in Q \cup P$ e $a \in T \cup \{\lambda\}$, δ é definida por

$$\delta(A, a) = \begin{cases} \{B \in V : A \to aB \in \widehat{P} \text{ ou } A \to Ba \in \widehat{P}\} & \text{, se } A \to a \notin \widehat{P} \\ \{B \in V : A \to aB \in \widehat{P} \text{ ou } A \to Ba \in \widehat{P}\} \cup \{C\} & \text{, se } A \to a \in \widehat{P} \end{cases}$$

Claramente, $L(M) = L(G)$. □

Note que os algoritmos nos teoremas 4.1 e 4.2 são duais, no sentido que aplicando um e em seguida o outro sempre voltaremos ao original.

Observe também que, o mecanismo de parada dos ALN, apesar de não explícitado em sua formulação matemática, pode ser formalizado através da noção de DI como segue: quando for achado uma DI (q, λ) apartir do movimento de uma outra DI (por exemplo, $(q', a) \vdash (q, \lambda)$), estamos na situação em que "uma cabeça leitora cruza a outra" e portanto a execução do ALN pára.

5. λ-Movimentos são Desnecessários

Um λ-ALN sem λ-transições será denominado de **autômato linear não-determinístico** (ALN).

Os λ-movimentos nas classes usuais de autômatos não são essenciais para os modelos. Por exemplo, λ-movimentos não aumentam o poder computacional de aceitação de autômatos finitos (veja, por exemplo, [14]), de autômatos com pilhas (veja, por exemplo, [12]) nem de máquinas de Turing (pela tese de Church). Assim, resulta razoável esperar que ALN e λ-ALN tenham exatamente o mesmo poder de aceitação. Porém, quando uma λ-transição em um λ-ALN ocorre entre dois estados de diferentes tipos não é óbvio como podemos eliminá-la sem alterar a linguagem aceita pelo autômato.

Teorema 5.1. *Seja $M = \langle Q, P, \Sigma, \delta, I, F\rangle$ um* λ-ALN. *Então existe um* ALN M' *tal que $\mathcal{L}(M) = \mathcal{L}(M')$.*

Demonstração. O seguinte algoritmo mostra como obter tal ALN M':

1. Adicione a I qualquer estado q para o qual exista uma λ-transição de um estado $q_0 \in I$ a q. Repetir este passo até não conseguir adicionar qualquer novo estado a I.

2. Para cada λ-transição de um estado $q' \in Q \cup P$ a um estado $q'' \in Q \cup P$:

   (a) Apague-a do diagrama de transição, e

   (b) Para cada estado $q \in Q \cup P$ e $a \in \Sigma \cup \{\lambda\}$, se há uma transição rotulada por a desde q a q', então adicione uma nova transição, também rotulada por a, de q a q''.

□

6. Aplicação

Linguagens formais têm diversas aplicações que vão além do desenvolvimento de compiladores. Por exemplo, podem ser aplicados em criptografia [24], em processamento digital de imagens [15], em biologia molecular [2], em teoria da optimalidade [18], em reconhecimentos de gestos da LIBRAS [6], etc. Aqui apresentaremos uma aplicação dos λ-ALN em bioinformática.

6.1. Palíndromos

Palíndromos, ou seja cadeias que têm o mesmo resultado quando lidas da esquerda para a direita ou da direita para esquerda, embora aparentem ser meros jogos linguísticos ou uma construção interessante em teorias dos números (por exemplo, o estudo dos números primos que são palíndromos em sua representação decimal [10]) têm aplicações importantes no estudo de sequências de DNA. A noção de palíndromos sobre um alfabeto Σ pode ser generalizada por considerar uma função $\phi : \Sigma \to \Sigma$ da seguinte forma: λ é um ϕ-palíndromo e $a \in \Sigma$ é um ϕ-palíndromo somente se $a = \phi(a)$. Se $w \in \Sigma^*$ é um ϕ-palíndromo então $aw\phi(a)$ também será um ϕ-palíndromo, para todo $a \in \Sigma$.

Uma função $\psi : \Sigma^* \to \Sigma$ é uma involução mórfica (antimórfica) se é involutiva, ou seja $\psi \circ \psi = Id_{\Sigma^*}$ e para todo $u, v \in \Sigma^*$, $\psi(uv) = \psi(u)\psi(v)$ $(\psi(uv) = \psi(v)\psi(u))$ [16, 17].

Proposição 6.1. *Seja $\phi : \Sigma \to \Sigma$ uma involução. Então, as funções $\widetilde{\phi}, \widehat{\phi} : \Sigma^* \to \Sigma^*$ definidas recursivamente por*

$$\widetilde{\phi}(\lambda) = \widehat{\phi}(\lambda) = \lambda, \quad \widetilde{\phi}(wa) = \widetilde{\phi}(w)\phi(a) \text{ e } \widehat{\phi}(wa) = \phi(a)\widehat{\phi}(w)$$

são, respetivamente, involuções mórficas e antimórficas.

Demonstração. Trivial. □

Claramente, $\widehat{\phi}(w) = \widetilde{\phi}(w)^R = \widetilde{\phi}(w^R)$, onde w^R é a cadeia reversa de w.

Teorema 6.2. *Seja $\phi : \Sigma \to \Sigma$ uma involução e $w \in \Sigma^*$. Então w é um ϕ-palíndromo sss $w = \widehat{\phi}(w)$*

Demonstração. A base da indução (λ e $a \in \Sigma$ tal que $a = \phi(a)$) é trivial. Como hipótese indutiva considere um ϕ-palíndromo $w \in \Sigma^*$ tal que $w = \widetilde{\phi}(w)$. Então,
$\widetilde{\phi}(aw\phi(a)) = \widetilde{\phi}(w\phi(a))\widetilde{\phi}(a) = \widetilde{\phi}(\phi(a))\widetilde{\phi}(w)\phi(a) = aw\phi(a)$. □

Em [16, 17] é usada a segunda parte deste teorema como definição primitiva de θ-palíndromo, onde $\theta : \Sigma^* \to \Sigma^*$ é uma função involutiva e antimórfica. Porém, nas provas de resultados em [17] usam de fato $\widehat{\phi}$. Um desses resultados é a Proposição 7 e o Lema 9, onde provam que dado uma involução antimórfica θ sobre um alfabeto Σ, o conjunto dos θ-palíndromos, P_θ, é uma linguagem livre de contexto não regular. Mas aqui refinaremos mais esse resultado.

Teorema 6.3. *Seja $\phi : \Sigma \to \Sigma$ uma involução. Então P_ϕ é uma linguagem linear (determinística).*

Demonstração. Seja $\Delta = \{a \in \Sigma : a = \phi(a)\}$. Então o ALN $M = \langle \{q_0\}, P, \Sigma, \delta, \{q_0\}, F \rangle$ tal que $P = \{p_a : a \in \Sigma\}$, $F = \{q_0\} \cup \{p_a : a \in \Delta\}$, e para todo $a \in \Sigma$, $\delta(q_0, a) = \{p_a\}$, $\delta(p_a, b) = \{q_0\}$ se $b = \phi(a)$, e $\delta(p_a, b) = \emptyset$ em outro caso. □

6.2. DNA e Palíndromos

Os ácidos nucléicos fazem parte das células vivas e são responsáveis pelo armazenamento e transmisão da informação genética assim como pela síntese precisa de proteínas. Os ácidos nucléicos são as biomoléculas que contém as informações genéticas e portanto são fundamentais para o controle celular. Elas estão compostas de subunidades chamadas de nucleotídeos (ácido fosfórico, açúcar, base purínica e base pirimidínica). Existem dois tipos de ácidos nucléicos: ácidos desoxirribonucléicos (DNA) e ácidos ribonucléicos (RNA). Uma molécula de DNA está formada por uma pentose (açúcar formado por 5 carbonos), um ácido fosfórico e uma base nitrogenada. Há quatro possíveis tipos de bases nitrogenadas no DNA - Adenina (A), Citosina (C), Guanina (G) e Timina (T) - as quais podem ser classificadas em purínicas (A e G) e pirimídicas (C e T). Estas quatro bases ligam-se ao açúcar e ao fosfato para formar DNA completo numa estrutura de dupla hélice [1]. Cada hélice forma uma cadeia (sequência) de bases nitrogenadas, de tal forma que há um emparelhamento de ambas, conectando cada base de uma cadeia com uma base da outra cadeia, fazendo uma espécie de ponte entre ambas as cadeias. Estas ligações só podem ser feitas entre bases do tipo A com T e do tipo C com G, o que leva a tratar essas bases como complementares.

Num nível mais abstrato, podemos considerar as bases nitrogenadas como um alfabeto $\Sigma = \{A, C, G, T\}$. A função $\theta : \Sigma \to \Sigma$ definida por $\theta(A) = T$, $\theta(T) = A$, $\theta(C) = G$ e $\theta(G) = C$ é claramente involutiva. θ é conhecida como a involução de Watson-Crick, por terem sido eles (James Watson e Francis Crick) que propuseram em 1953 o modelo de dupla hélice para o DNA e portanto esse aparelhamento entre bases complementares. Um θ-palíndromo é conhecido como palíndromo de Watson-Crick [17].

θ-palíndromos e outros tipos de redundâncias ocorrem com frequências em subcadeias de DNA (veja por exemplo [21]). Uma questão que torna importante o estudo destas sequências é que alguma enzimas têm a capacidade de reconhecer palíndromos para depois removê-los, provocando uma mutação no DNA. Neste sentido, como demonstrado no Teorema 6.3, os λ-ALN podem ser uma ferramenta útil para esse tipo de pesquisa. Por exemplo, o λ-ALN da Figura 2 é capaz de detectar se um DNA contém um segmento que seja um θ-palíndromo de tamanho superior ou igual a $2n$, para algum $n \geq 3$, mesmo que suas metades estejam separadas por diversas bases nitrogenadas.

7. Considerações Finais

Este trabalho introduz um modelo de autômatos intuitivo e simples para linguagens lineares que estende de maneira natural os autômatos finitos não-determinísticos com λ-transições (λ-AFN). Mais ainda, os λ-ALN são intrínsecos, no sentido que não consideram agentes externos ao modelo, como ocorre com os outros modelos de autômatos para esta classe de linguagens. Outra propriedade interessante dos λ-ALN, que não possuem os outros modelos para linguagens lineares, é que eles

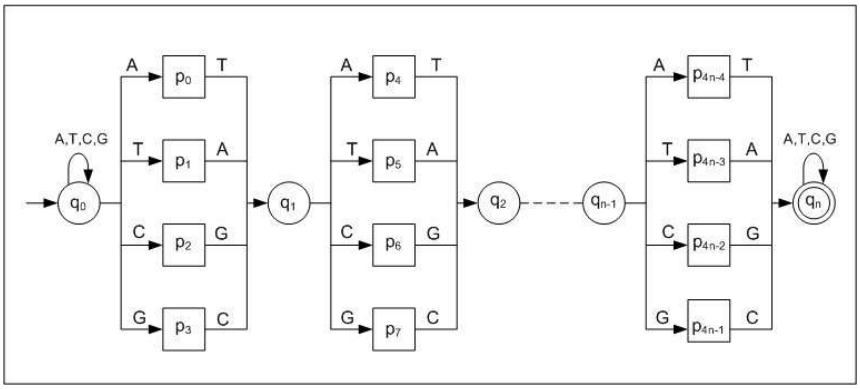

Figura 2: λ-ALN que detecta em um DNA a ocorrência de algum θ-palindromo de tamanho maior ou igual a 2n.

estendem de modo natural λ-AFN, no sentido que todo λ-AFN é um λ-ALN com $P = \emptyset$.

Neste artigo também provamos que, analogamente ao caso de autômatos finitos, autômatos com pilha e máquinas de Turing, os λ-movimentos não acrescentam poder computacional a λ-ALN.

Como uma forma de mostrar a utilidade deste novo formalismo, apresentamos uma aplicação em sequências palindrômicas de DNA. De passo, apresentamos novos resultados teóricos nesta área.

A simplicidade das provas neste artigo advem da simplicidade do modelo o qual seria outra vantagem deste modelo quando comparado com os outros modelos de autômatos para a classe das linguagens lineares.

Como trabalhos futuros pretende-se estudar algumas subclases de ALN, como por exemplo autômatos lineares determinísticos (ALD) e comparar a classe de linguagens aceitas por ALD com algumas das classes de linguagens lineares apresentadas em [9] e que são chamadas de "determinísticas". Em seguida poderiamos estabelecer uma hierarquia infinita de classes de linguagens iniciando dessa classe de linguagens lineares "determinísticas" até a classe das linguagens lineares, de forma análoga à hierarquia de linguagens livres de contexto em [4].

Agradecimentos

Agradeço aos revisores anônimos pelas suas sugestões que ajudaram a melhorar a qualidade do artigo assim como ao meu aluno de doutorado, Anderson de Paiva Cruz, pela revisão final e sugestões feitas.

Abstract. In this paper we introduce the class of linear non-deterministic automata with λ-transitions, and based on a new normal form for linear grammars, we prove that the class of languages accepted by such automata is exactly the class of linear languages. Also we show that, analogously to finite automata and pushdown automata, the ocorrences of λ-transitions in a non-deterministic linear automaton not means that not be possible define a non-deterministic linear automata without λ-transitions which accept the same language. Thus, λ-transitions not increases the acceptance power of this class of automata and therefore can be dispensed. Finally, we present an application of this automata in the context of bioinformatics.

Referências

[1] B. Albert, "Molecular Biology of the Cell", 4a ed., Garland Science, New York, 2002.

[2] J.A. Anderson, "Automata Theory with Modern Applications", Cambridge University Press, Cambridge, 2006.

[3] J.M. Autebert, J. Berstel, L. Boasson, Context-free languages and pushdown automata, em "Handbook of Formal Languages, Vol. 1: Word, Language, Grammar" (G. Rozenberg, ed.), pp. 111–174, Springer-Verlag, New York, 1997.

[4] B.C. Bedregal, Pushdown automata free of explicit nondeterminism and an infinite hierarchy of context-free languages, *Fundamenta Informaticae*, **81** (2007), 367–377.

[5] B.C. Bedregal, B.M. Acióly, A. Lyra, "Introdução à Teoria da Computação: Linguagens Formais, Autômatos e Computabilidade", Editora UnP, Natal, 2010.

[6] B.C. Bedregal, A.C.R. Costa, G.P. Dimuro, Fuzzy rule-based hand gesture recognition, *IFIP Int. Fed. for Information Processing*, 217, pp. 285–294, 2006.

[7] J. Carroll, D. Long, "Theory of Finite Automaton With an Introduction to Formal Languages", Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

[8] D.I.A. Cohen, "Introduction to Computer Theory" (Revised printing), John Wiley & Sons, New York, 1991.

[9] C. De La Higuera, J. Oncina, Inferring Deterministic Linear Languages, em "Proc. of the 15^{th} Annual Conference on Computational Learning Theory (DLT)"(J. Kivinen, R.H. Sloan, eds.), Lectures Notes in Computer Science, Vol. 2375, pp. 185–200, Springer-Verlag, Berlim, 2002.

[10] H.T.A. De Sena, Algumas Evidencias Computacionais da Infinitude dos Números Primos Palindrômicos e Generalizações Destes. Trabalho de Conclusão de Curso. (Graduação em Bach. em Ciências da Computação), Universidade Federal do Rio Grande do Norte, Natal, 2008. (Disponível em http://www.dimap.ufrn.br/∼bedregal/students.html)

[11] S. Ginsburg, E.H. Spanier, Finite-turn pushdown automata. *SIAM Journal on Computing*, **4** (1966), 429–453.

[12] M.A. Harrison, "Introduction to Formal Language Theory", Addison-Wesley, Reading, MA, 1978.

[13] H.J. Hoogeboom, J. Engelfriet, Pushdown automata, em "Formal Languages and Applications" (C. Martin-Vide, V. Mitrana, G. Paun, eds), Studies in Fuzziness and Soft Computing Series, pp. 117–138, Springer, Berlim, 2004.

[14] J.E. Hopcroft, J.D. Ullman, "Introduction to Automata Theory, Languages and Computation", Addison-Wesley, Reading, MA, 1979.

[15] J. Kari, Image processing using finite automata, em "Recent Advances in Formal Languages and Applications", (Z. Ésik, C. Martín-Vide, V. Mitrana, eds.), Studies in Computational Intelligence Series, Vol. 25, pp. 171–208, Springer-Verlag, Berlim, 2006.

[16] L. Kari, K. Mahalingan, Involutive bordered words, *Int. J. of Foundations of Computer Sviences*, **18**, No. 5 (2007), 1089–1106.

[17] L. Kari, K. Mahalingan, Watson-Crick palindromes in DNA computing, *Natural Computing*, **9** (2010), 297–316.

[18] S. Kepser, U. Mönnich, Closure properties of linear context-free languages with an application to optimality theory, *Theoretical Computer Science*, **354** (2006), 82–97.

[19] P. Linz, "An Introduction to Formal Language and Automata", Jones and Bartlett Publisher, Sudbury, MA, 2001.

[20] E. Mäkinen, Inferring finite transducer, *Journal Brazilian Computational Society*, **9**, No. 1 (2003), 5–8.

[21] M. Okuno, E. Arimoto, Y. Ikenobu, T. Nishihara, M. Imagawa, Dual DNA-binding specificity of peroxisome-proliferator-activated receptor γ controlled by heterodimer formation with retinoid X receptor α, *Biochemistry Journal*, **353** (2001), 193–198.

[22] A.L. Rosenberg, A machine realization of the linear context-free languages, *Information and Control*, **10** (1967), 175–188.

[23] A. Salomaa, "Formal Languages", Computer Science Classics Series, Academic Press, 1973.

[24] R. Tao, "Finite Automata and Application to Cryptography", Springer, Berlin, 2009.

[25] N.J. Vieira, "Introdução aos Fundamentos da Computação: Linguagens e Máquinas", Thomson, São Paulo, 2006.