Algoritmo de Earley Adaptativo para Reconhecimento de Padrões Sintáticos

G. P. Santos[1], J. J. Neto[2] e A. Machado-Lima[3]

***Resumo*—A teoria das linguagens formais é amplamente utilizada nos processos de solução de problemas de naturezas diversas, como por exemplo reconhecimento de padrões sintáticos. Gramáticas adaptativas, em particular, representam soluções eficientes quando se necessita incluir na modelagem dependências de contexto. Embora existam eficientes analisadores sintáticos para essas gramáticas, a grande maioria identifica apenas uma árvore sintática de uma dada cadeia ou exige que a gramática esteja em alguma forma normal. No entanto, há problemas para os quais se necessita obter todas as árvores de derivação e/ou utilizar gramáticas sem restrições aos formatos das produções. Como exemplo podemos citar a modelagem de famílias de RNAs com estrutura secundária. A fim de preencher essa lacuna, este artigo propõe uma versão adaptativa do algoritmo analisador sintático de Earley. Tal implementação fará parte de um arcabouço de geração de classificadores baseados em gramáticas que está sendo aprimorado com a incorporação de gramáticas adaptativas. A adaptatividade será uma importante ferramenta no tratamento do problema de caracterização de famílias de RNAs contendo pseudonós.**

***Palavras-chaves*—Reconhecimento de Padrões, Métodos Sintáticos, Métodos Adaptativos, Gramáticas, Earley**

I. Introdução

A teoria das linguagens formais, elaborada com o objetivo de desenvolver teorias relacionadas a linguagens naturais, logo passou a ser notada como importante também para o estudo das linguagens artificiais [1]. Assim, as linguagens formais passaram a ser utilizadas amplamente na análise sintática de linguagens de programação, na modelagem de circuitos e redes lógicas, em sistemas biológicos, sistemas de animação, hipertexto, linguagens não lineares e outros [1].

O conjunto de cadeias que compõem uma linguagem pode ser exaustivamente sintetizado por gramáticas, que são sistemas formais baseados em regras de substituição [2]. Para cada classe de linguagem há um tipo de gramática capaz de a representar, seguindo a hierarquia de Chomsky [3], que é organizada de acordo com a complexidade do formalismo.

As gramáticas podem ser utilizadas na área de reconhecimento de padrões sintáticos, uma vez que podem modelar a hierarquia dos componentes que formam as cadeias de uma linguagem, podendo ser traçado um paralelo entre esta hierarquia e a decomposição de padrões em subestruturas [4]. Assim, a árvore de derivação, ou árvore sintática, de uma gramática representa um padrão que pode ser utilizado para classificação.

Gramáticas podem também ser utilizadas no contexto de aprendizado estatístico, sendo neste caso utilizadas suas versões estocásticas. Uma gramática estocástica é uma gramática em que as regras de produção $P \subset \{\alpha \rightarrow \beta, p\}$, sendo $\alpha \in V^*NV^*$, $\beta \in V^*$, V o conjunto finito e não vazio de símbolos que representam o vocabulário da gramática e N o conjunto de símbolos não terminais da gramática, e p um valor de probabilidade, $0 \leq p \leq 1$, de tal forma a definir uma distribuição de probabilidades sobre as produções com o mesmo lado esquerdo, ou seja, considerando as produções $\alpha \rightarrow \beta_i, p_i \in P$, $\sum_i p_i = 1$.

As gramáticas estocásticas, ao invés de determinar se uma dada cadeia pertence à linguagem gerada pela gramática, atribui a ela uma probabilidade de pertencer à linguagem, ou seja, de ser gerada pela gramática. Essa probabilidade consiste no produto das probabilidades de todas as produções utilizadas na derivação da árvore sintática da cadeia. Se a gramática for ambígua, todas as árvores sintáticas da cadeia devem ser consideradas no cálculo desta probabilidade, somando-se as probabilidades dadas por cada árvore. Além disso, para gramáticas ambíguas, também é possível determinar qual a árvore sintática mais provável para uma dada cadeia, isto é, qual o padrão sintático mais provável associado à sua geração.

Considerando várias gramáticas estocásticas G_i, cada uma representando um padrão sintático, um classificador pode ser definido da seguinte forma: dada uma cadeia c, classifica-se a cadeia c como pertencente à linguagem que é gerada pela gramática G_K, sendo $K = \text{argmax}_i\ P(c|G_i)$.

A utilização de gramáticas estocásticas exemplifica a necessidade de algoritmos de análise sintática que sejam capazes de encontrar não só uma, mas todas as árvores sintáticas de uma cadeia quando a gramática em questão for ambígua.

A fim de facilitar a utilização de gramáticas em reconhecimento de padrões sintáticos, foi desenvolvido um arcabouço de geração classificadores baseados em gramáticas estocásticas, chamado GrammarLab.

Um dos problemas no quais o GrammarLab está sendo utilizado é o problema de Bioinformática de caracterização de famílias de RNAs. Tal caracterização, englobando tanto sequência quanto estrutura, exige gramáticas no mínimo livres de contexto. As regras de produção devem possuir um formato adequado para modelar os pareamentos existentes entre bases nucleotídicas do RNA, não podendo estar limitadas, por exemplo, à forma normal de Chomsky. Por isso o GrammarLab utiliza o analisador sintático de Earley [5], já que ele não assume que a gramática de entrada esteja em nenhuma forma normal e é capaz de encontrar todas as árvores sintáticas de uma cadeia. Mas para uma modelagem que inclua certos padrões estruturais conhecidos como pseudonós, é necessária a inclusão de dependência de contexto. Os pseudonós, por apresentarem relações de dependências cruzadas (Figura 1), tornam necessário o uso de gramáticas sensíveis ao contexto [6]. No

[1]EACH - Universidade de São Paulo - gilmar.santos@usp.br
[2]POLI - Universidade de São Paulo - joao.jose@poli.usp.br
[3]EACH - Universidade de São Paulo - ariane.machado@usp.br

entanto, o GrammarLab não possui suporte para gramáticas sensíveis ao contexto devido à alta complexidade de tempo da análise sintática dessa classe de gramáticas, cujo problema geral é NP-completo [7] [8] [9].

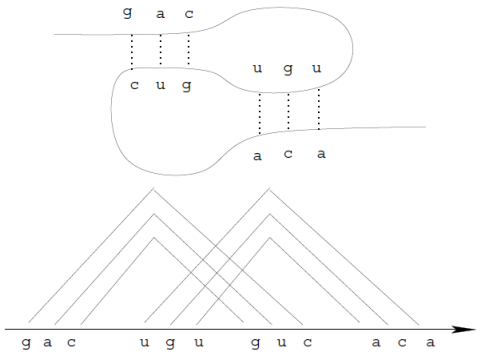

Figura 1. Estrutura e dependências cruzadas de um pseudonó.

A fim de sanar essa lacuna, este artigo apresenta uma proposta de evolução do algoritmo de Earley com a inclusão de adaptatividade em sua constituição, de tal forma que suas características como analisador sintático de gramáticas livres de contexto sejam expandidas para reconhecer gramáticas adaptativas, que por sua vez são capazes de modelar trechos dependentes de contexto sem aumentar excessivamente o tempo de análise.

Este é o primeiro passo de uma série no aprimoramento do GrammarLab, sendo que a partir desta melhoria o arcabouço será capaz de lidar com o problema de classificação de sequências de RNAs mesmo de famílias cujas estruturas apresentem elementos que necessitem de uma gramática no mínimo sensível ao contexto, sem no entanto cair em um problema NP-completo.

O restante deste artigo está organizado da seguinte forma. A seção II traz os conceitos básicos para o entendimento deste trabalho, a saber sobre analisadores sintáticos para gramáticas livres de contexto, o algoritmo analisador de Earley e dispositivos adaptativos. A seção III apresenta brevemente a atual versão do GrammarLab. A seção IV apresenta a proposta do algoritmo de Earley adaptativo. Finalmente, a seção V traz as conclusões deste trabalho.

II. Conceitos fundamentais

A. Analisadores Sintáticos

Para cada gramática livre de contexto pode ser desenhado um autômato a pilha que reconheça a linguagem gerada por tal gramática [10], sendo a complexidade de reconhecimento de ordem polinomial. Dada sua complexidade de reconhecimento e a capacidade de gerar todas as linguagens regulares e muitas linguagens adicionais [10], as gramáticas livres de contexto acabam se tornando um interessante objeto de estudo.

Outra alternativa de reconhecimento das linguagens livres de contexto são os algoritmos analisadores sintáticos de gramáticas, que diferem dos autômatos a pilha por serem de propósito geral, isto é, não serem desenhados para operar em uma linguagem livre de contexto específica. Dentre os algoritmos analisadores de gramáticas livres de contexto, podemos destacar o algoritmo CYK [11] e o algoritmo de Earley [5].

O algoritmo CYK utiliza uma estratégia de análise de baixo para cima (*bottom-up parser*) e programação dinâmica, resolvendo o problema de reconhecimento e determinação das diferentes árvores sintáticas com uma complexidade de tempo $O(n^3)$, sendo n o tamanho da sequência testada. Uma desvantagem do algoritmo CYK é a necessidade da gramática estar representada na forma normal de Chomsky. Uma gramática está na forma normal de Chomsky quando todas as suas regras de produção são da forma:

- $A \rightarrow BC, A \in N, B \in N, C \in N$
- $A \rightarrow \alpha, A \in N, \alpha \in \Sigma$

sendo N o conjunto de símbolos não terminais da gramática e Σ o conjunto finito e não vazio de símbolos que representa o alfabeto, os símbolos terminais, da gramática.

Toda gramática pode ser convertida para a forma normal de Chomsky [10]. No entanto, essa normalização pode afetar a informação estrutural fornecida pelas árvores sintáticas da gramática original.

B. Algoritmo de Earley

Diferentemente do algoritmo CYK, o algoritmo de Earley não necessita que a gramática esteja em uma forma específica, eliminando a necessidade de adaptação da gramática ou processamento adicional para normalizar uma gramática que se deseja analisar [5]. Além disso, o algoritmo de Earley [5] apresenta a característica de identificar as diferentes árvores de derivação, possuindo uma complexidade computacional de ordem $O(n^3)$ para gramáticas ambíguas. Isso faz com que seja um algoritmo conveniente, por exemplo, para o uso em problemas de bioinformática relacionados com análise de sequências biológicas, que por natureza são linguagens ambíguas, e que demandam que o formato das regras de produção mimetizem a estrutura de pareamentos entre as bases nucleotídicas (símbolos terminais na gramática).

Basicamente, o algoritmo de Earley percorre uma dada cadeia de entrada $X_1...X_n$ da esquerda para a direita. Para cada símbolo X_i percorrido, um conjunto de estados S_i é construído, representando a condição do processo de reconhecimento no ponto atual. Cada estado $s \in S_i$ é um ítem de análise que é representado por uma quádrupla, a qual é formada por: (i) uma produção da gramática; (ii) uma posição que indica até que ponto desta regra o reconhecimento foi realizado com sucesso; (iii) um ponteiro para a posição na cadeia de entrada em que se iniciou a busca pela instância da produção e (iv) uma cadeia formada pelos próximos k símbolos terminais seguintes à produção (geralmente é considerado apenas um).

Cada estado s é utilizado no desenvolvimento da análise sintática por meio da aplicação de uma dentre três possíveis operações do algoritmo: (i) *predictor*, que é a operação na qual são expandidos os símbolos não terminais das regras de produção, adicionando novos estados no conjunto S_i atual; (ii) *scanner*, operação na qual é verificado se o símbolo terminal sendo analisado da regra de produção do estado atual é o

símbolo X_i atual, gerando um novo conjunto de estados S_{i+1} e seguindo para o próximo símbolo de entrada X_{i+1} e (iii) *completer*, operação aplicada quando a produção do estado s foi analisada completamente, sendo comparados os k símbolos candidatos sucessores aos símbolos de entrada $X_{i+1}..X_{i+k}$, adicionando ao conjunto S_i atual os estados com as produções nas quais o símbolo completado aparece do lado direito.

Ao finalizar a varredura da cadeia de entrada, o algoritmo identifica se a cadeia pertence ou não à linguagem descrita pela gramática em questão e, caso afirmativo, pode retornar todas as árvores de derivação (utilizando os ponteiros dos estados para navegar pelo resultado da análise sintática).

C. Dispositivos Adaptativos

O uso de métodos adaptativos [2] possibilita alterar dinamicamente as propriedades de uma gramática, viabilizando inserir sensibilidade ao contexto em uma gramática originalmente livre de contexto sem, com isso, aumentar excessivamente sua complexidade de análise. Isso é possível pois as regiões dependentes de contexto de uma cadeia podem ser confinadas em regiões controladas. Ou seja, é realizada uma separação da cadeia em regiões localmente regulares ou livres de contexto, regiões estas posteriormente integradas em uma análise global dependente de contexto. Tal separação de análise pode resultar em uma economia considerável de tempo de reconhecimento.

Dispositivos adaptativos são dispositivos formais que podem ter seu comportamento alterado de forma dinâmica como resposta espontânea a estímulos de entrada [12].

Quaisquer alterações possíveis no comportamento de um dispositivo adaptativo devem ser conhecidas *a priori*. Assim, esses dispositivos são capazes de detectar as situações que disparam as modificações e devem ser automodificáveis para reagir de forma adequada, se adaptando à situação.

Um dispositivo adaptativo é formado pela incorporação de ações adaptativas às regras de um dispositivo não adaptativo subjacente. Assim, sempre que alguma dessas regras é aplicada, a ação adaptativa correspondente é acionada. Dessa forma, o dispositivo adaptativo resultante pode ser facilmente compreendido por todos que tenham familiaridade com o dispositivo subjacente.

Em [13] é apresentado um formalismo para uma gramática adaptativa sensível ao contexto e é estabelecida uma equivalência com autômatos adaptativos [14].

D. Gramática Adaptativa

Uma gramática adaptativa é um dispositivo adaptativo, conforme descrito em [13]. Nesta pesquisa, a gramática utilizada será representada formalmente por $G_A = (G^0, T, R^0)$, na qual:

- G^0 é a gramática livre de contexto inicial;
- T é um conjunto finito, possivelmente vazio, de funções adaptativas, que são conjuntos de ações adaptativas [12];
- R^0 é a relação entre as regras de produção da gramática G^0 e as funções adaptativas, sendo $R^0 \subseteq BA \times P^0 \times AA$;
- $BA \subseteq (T \cup \{\epsilon\})$ é o conjunto de ações adaptativas executadas antes do acionamento da respectiva regra de produção $p \in P^0$;
- $AA \subseteq (T \cup \{\epsilon\})$ é o conjunto de ações adaptativas executadas após o acionamento da respectiva regra de produção $p \in P^0$.

sendo P^0 o conjunto de produções da gramática G^0 e $\{\epsilon\}$ o conjunto que tem como único elemento ϵ, que representa uma ação adaptativa nula.

A cada ação adaptativa executada, uma nova gramática livre de contexto G^i é gerada.

Neste trabalho será considerado como dispositivo subjacente uma gramática livre de contexto, de forma a reduzir, no geral, a complexidade de análise.

Exemplo de Gramática Adaptativa

O seguinte exemplo, utilizando formalismo adaptativo [13] [12], apresenta uma gramática simplificada que representa a linguagem $L = a^n b^m c^n d^m$, que representa uma dependência cruzada na qual há uma relação entre os símbolos terminais a e c e entre os símbolos b e d.

$$
\begin{array}{l}
G_A = (G^0, T, R^0) \\
G^0 = \{V^0, \Sigma, P^0, S\} \\
V^0 = \{S, K, a, b, c, d\} \\
\Sigma = \{a, b, c, d\} \\
\\
P^0 = \{ \\
\quad S \rightarrow K\{AdP(K)\} \\
\quad K \rightarrow \phi \\
\} \\
\\
T = \{ \\
\quad AdP(X) = \{A'*, B'*, C'*, D'* : \\
\qquad +[X \rightarrow A'B'C'D'] \\
\qquad +[A' \rightarrow aA'\{Ad1(A', C', a, c)\}] \\
\qquad +[A' \rightarrow a\{Ad2(C', c)\}] \\
\qquad +[B' \rightarrow bB'\{Ad1(B', D', b, d)\}] \\
\qquad +[B' \rightarrow b\{Ad2(D', d)\}] \\
\qquad +[C' \rightarrow \phi] \\
\qquad +[D' \rightarrow \phi] \\
\quad \} \\
\\
\quad Ad1(N, X, x, y) = \{X'* : \\
\qquad -[N \rightarrow xN\{Ad1(N, X, x, y)\}\}] \\
\qquad -[N \rightarrow x\{Ad2(X, y)\}] \\
\qquad +[N \rightarrow xN\{Ad1(N, X', x, y)\}\}] \\
\qquad +[N \rightarrow x\{Ad2(X', y)\}] \\
\qquad +[X \rightarrow yX'] \\
\qquad +[X' \rightarrow \phi] \\
\quad \} \\
\\
\quad Ad2(X, y) = \{ \\
\qquad +[X \rightarrow y] \\
\quad \} \\
\}
\end{array}
$$

Neste exemplo, a regra de produção inicial $S \rightarrow K$ possui uma associação com uma função adaptativa $\{AdP(K)\}$, sendo

que a produção $K \rightarrow \phi$ indica que o símbolo não terminal K será definido de forma dinâmica, pelo acionamento de alguma ação adaptativa. A função adaptativa $AdP(X)$, ao ser acionada, basicamente indica o início de um domínio de dependência cruzada desmembrado em quatro partes, sendo A^i relacionado com C^i e B^i relacionado com D^i. Cada vez que uma regra de produção associada com A^i ou B^i é aplicada, uma nova regra associada a C^i e/ou D^i é adicionada ao conjunto de regras de produção, forçando a dependência cruzada.

Abaixo, uma derivação de uma dependência cruzada utilizando a gramática adaptativa de exemplo, e na Figura 2 o destaque para a dependência apresentada.

$$
\begin{aligned}
S &\Rightarrow_{G^0} K \\
&\Rightarrow_{G^1} A^1B^1C^1D^1 \\
&\Rightarrow_{G^1} aA^1B^1C^1D^1 \\
&\Rightarrow_{G^2} aaB^1C^1D^1 \\
&\Rightarrow_{G^3} aabB^1C^1D^1 \\
&\Rightarrow_{G^4} aabbB^1C^1D^1 \\
&\Rightarrow_{G^5} aabbbC^1D^1 \\
&\Rightarrow_{G^6} aabbbcC^2D^1 \\
&\Rightarrow_{G^6} aabbbccD^1 \\
&\Rightarrow_{G^6} aabbbccdD^2 \\
&\Rightarrow_{G^6} aabbbccddD^3 \\
&\Rightarrow_{G^6} aabbbccddd
\end{aligned}
$$

$$\Rightarrow_{G^6} aabbbccddd$$

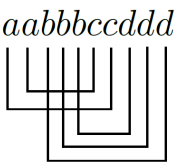

Figura 2. Dependência cruzada.

III. GrammarLab: Laboratório de Geração de Classificadores

O GrammarLab [15] é um arcabouço desenvolvido em C++ com o objetivo de facilitar a implementação e geração de classificadores baseados em gramáticas estocásticas, facilitando a implementação de algoritmos de inferência gramatical, de estimação das probabilidades para tornar uma gramática estocástica, de geração de analisadores sintáticos e de combinação dos mesmos para gerar os classificadores. Na Figura 3 é apresentada uma visão geral do arcabouço sendo utilizado para classificação de padrões sintáticos.

O arcabouço é composto por três partes: (i) algoritmos de inferência gramatical e estimação de probabilidades, (ii) módulo de suporte a implementação e (iii) módulo de suporte a testes.

A parte (i) é constituída de classes abstratas e concretas de inferidores gramaticais e de estimadores de probabilidades.

A parte (ii) é constituída de uma biblioteca de classes que implementam estruturas de dados úteis nesse contexto, como gramáticas, autômatos a árvore, trie e conjuntos de classes que modelam *streams* de entrada e saída que possibilitam um

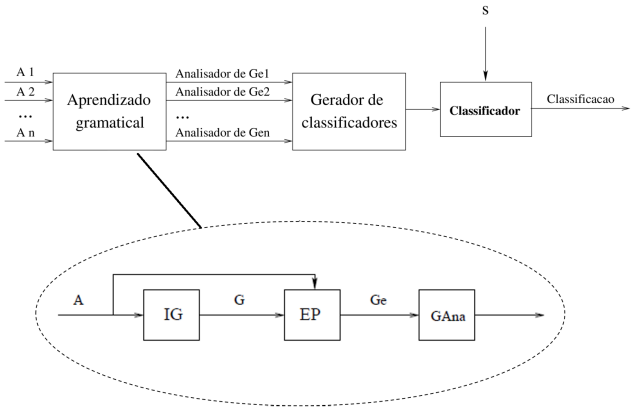

Figura 3. GrammarLab: visão geral.

canal de comunicação entre os algoritmos. Também são encontradas classes para geração de analisadores sintáticos e um mecanismo integrador de todos os analisadores das gramáticas consideradas na classificação. O algoritmo de Earley [5] foi escolhido como implementação para o analisador sintático de gramáticas livres de contexto estocásticas no arcabouço pelo fato de fornecer todas as árvores de derivação e por não exigir nenhuma normalização específica da gramática, conforme dito anteriormente.

A parte (iii) é formada por programas Perl e C++ que tratam a geração dos classificadores, execução de testes e obtenção de resultados.

Como abordagem para otimização de tempo de execução, o arcabouço utiliza uma estratégia de geração de códigos fontes (extensões .cpp e .h) para cada gramática inferida pelos algoritmos de aprendizado e seus analisadores sintáticos. Em contrapartida, tanto a gramática, quando os analisadores têm um comportamento totalmente estático.

IV. Proposta: GrammarLab Adaptativo

Uma vez que o arcabouço GrammarLab possui toda uma infraestrutura que atende às principais demandas para reconhecimento de padrões sintáticos para o problema alvo citado de caracterização de famílias de RNAs, como a implementação do algoritmo de Earley e várias classes de apoio para a implementação de classificadores, a proposta desse projeto é a evolução do arcabouço para que o mesmo possa lidar com gramáticas sensíveis ao contexto. Para isso, conforme visto na seção II-C, serão utilizados dispositivos adaptativos, de forma que a complexidade de análise sintática não seja aumentada excessivamente ao ponto de inviabilizar seu uso na prática.

A modelagem atual de classes C++ que representam uma gramática será alterada para permitir a inclusão de ações adaptativas associadas às regras de produção de uma gramática livre de contexto, seguindo o formalismo descrito em [12] e inspirado em [13].

O foco da primeira parte da pesquisa é a implementação da análise sintática baseada no algoritmo de Earley para gramáticas adaptativas, permitindo assim a classificação de sequências de estruturas que apresentam elementos que necessitem de uma gramática no mínimo sensível ao contexto, a exemplo dos

pseudonós encontrados nas estruturas secundárias de alguns RNAs.

O comportamento estático atual do GrammarLab precisará ser alterado para que possa comportar a alteração dinâmica nas regras de produções da gramática, comportamento característico dos dispositivos adaptativos.

A. Algoritmo de Earley Adaptativo

O algoritmo de Earley pode ser visto como um mapeamento de uma gramática livre de contexto em um conjunto de regras de análise R. Este conjunto de regras, por sua vez, pode ser considerado como descritor do dispositivo subjacente da adaptatividade.

Considerando que uma ação adaptativa A associada a uma produção P de uma gramática é executada sempre que P for ativada, temos que A é executada sempre que no conjunto R for ativado o grupo $R(P)$, ou seja, o conjunto de regras associado à ativação da produção P.

Se, ao ser especificada a gramática adaptativa proposta, cada uma de suas produções livres de contexto P for associada à respectiva ação adaptativa A, então será possível construir um a um os respectivos conjuntos de regras de análise $R(P)$ e, nessa mesma ocasião, poderá ser automaticamente acoplada a respectiva ação adaptativa A ao correspondente conjunto $R(P)$, de modo que, em tempo de execução, caso a produção P seja escolhida para ser aplicada, a ativação de A seja uma decorrência da ativação de $R(P)$, de forma que $P \rightarrow \{Apply(R(P)); Exec(A)\}$.

Assim, é necessária a alteração da operação *predictor* do algoritmo de Earley (descrito na seção II-B) para que seja aplicada a ação adaptativa e gerada uma nova gramática livre de contexto G^i, assim como a inclusão de um ponteiro para a gramática referenciada no controle de estados do algoritmo.

B. Exemplo de Aplicação do Algoritmo

A seguir, um exemplo de uma análise sintática considerando a gramática adaptativa e a cadeia de exemplo apresentadas na seção II-D.

Inicialmente, a cadeia de entrada é complementada com $k+1$ símbolos $\dashv$ não pertencentes ao alfabeto da gramática.

$$X = aabbbccddd \dashv\dashv$$

Passo $i = 0$, conjunto de estados S_0 e $X_1 = a$

Como passo inicial $i = 0$, o estado inicial é adicionado ao conjunto de estados S_0.

$$\phi \rightarrow .S \dashv \quad \dashv \;\; 0 \;\; G^0 \tag{1}$$

Nessa representação de um estado, $\phi \rightarrow S \dashv$ é a produção, o . indica até que ponto o reconhecimento já foi realizado com sucesso, o símbolo $\dashv$ representa a cadeia de símbolos terminais seguintes à produção, 0 é o ponteiro para a posição na cadeia de entrada em que se iniciou a busca pela instância da produção e G^0 representa a gramática resultante do acionamento da produção.

Como o símbolo após . é o símbolo não terminal S, a operação *predictor* é aplicada à produção inicial original da gramática G^0, ativando a função adaptativa associada com tal produção e gerando uma nova gramática G^1.

$$S \rightarrow .K\{AdP(K)\} \quad \dashv \;\; 0 \;\; G^1 \tag{2}$$

Novamente a operação *predictor* é aplicada ao novo estado, adicionando um novo estado no conjunto S_0, utilizando desta vez a produção da gramática G^1. Como não há uma função adaptativa associada com a produção, desta vez a gramática não evolui.

$$K \rightarrow .A^1B^1C^1D^1 \quad \dashv \;\; 0 \;\; G^1 \tag{3}$$

A operação *predictor* é aplicada mais uma vez. Como o símbolo seguinte a A^1 é o não terminal B^1, o mesmo é avaliado e são considerados os primeiros símbolos não terminais possíveis como a cadeia seguinte à produção, adicionando então os seguintes estados ao conjunto S_0.

$$A^1 \rightarrow .aA^1\{Ad1(A^1, C^1, a, c)\} \quad b \;\; 0 \;\; G^{2.1} \tag{4}$$

$$A^1 \rightarrow .a\{Ad2(C^1, c)\} \quad b \;\; 0 \;\; G^{2.2} \tag{5}$$

A operação *predictor* não pode ser aplicada a nenhum dos últimos estados adicionados, pois em todos eles o símbolo à direita do ponto é um símbolo terminal. Neste caso, a operação *scanner* é aplicada, adicionando ao conjunto de estados S_{i+1} os estados em que o símbolo terminal seguinte ao ponto for igual ao símbolo $X_{i+1} = a$ da cadeia de entrada. Os estados são copiados, tendo como alteração apenas a posição do ponto, que é deslocado um símbolo para a direita, indicando que o símbolo terminal anterior foi processado.

Assim, o conjunto de estados S_1, após a aplicação da operação *scanner* em todos os últimos estados de S_0, possui os seguintes estados:

$$A^1 \rightarrow a.A^1\{Ad1(A^1, C^1, a, c)\} \quad b \;\; 0 \;\; G^{2.1} \tag{6}$$

$$A^1 \rightarrow a\{Ad2(C^1, c)\}. \quad b \;\; 0 \;\; G^{2.2} \tag{7}$$

Passo $i = 1$, conjunto de estados S_1 e $X_2 = a$

Como não existem mais estados a serem processados no conjunto S_0, o algoritmo evolui para o passo $i = 1$ e o conjunto S_1 se torna o conjunto a ser processado.

Como o símbolo não terminal A^1 está após o ponto no estado 6, a operação *predictor* é aplicada, adicionando ao conjunto S_1 os estados:

$$A^1 \rightarrow .aA^1\{Ad1(A^1, C^2, a, c)\} \quad b \;\; 1 \;\; G^{3.1} \tag{8}$$

$$A^1 \rightarrow .a\{Ad2(C^2, c)\} \quad b \;\; 1 \;\; G^{3.2} \tag{9}$$

A operação *completer* é aplicada nos estados em que o ponto se encontra no fim da produção, sendo comparados os k símbolos da cadeia posterior do estado com os símbolos $X_{i+1}..X_{i+k}$ da cadeia de entrada. O estado 7 possui uma cadeia posterior divergente da cadeia de entrada, sendo portanto descartado.

Após a aplicação da operação *scanner* nos últimos estados de S_1, os seguintes estados são adicionados ao conjunto de estados S_2:

$$A^1 \rightarrow a.A^1\{Ad1(A^1, C^2, a, c)\} \quad b\ 1\ G^{3.1} \tag{10}$$

$$A^1 \rightarrow a\{Ad2(C^2, c)\}. \quad b\ 1\ G^{3.2} \tag{11}$$

O algoritmo evolui para o passo $i = 2$ na sequência.

Passo $i = 2$, conjunto de estados S_2 e $X_2 = b$

A operação *predictor* é aplicada ao estado 10, adicionando ao conjunto S_2 os estados:

$$A^1 \rightarrow .aA^1\{Ad1(A^1, C^3, a, c)\} \quad b\ 2\ G^{4.1} \tag{12}$$

$$A^1 \rightarrow .a\{Ad2(C^3, c)\} \quad b\ 2\ G^{4.2} \tag{13}$$

A operação *completer* é aplicada ao estado 11 e, como a cadeia posterior da produção é igual à cadeia de entrada $X_3 = b$ sendo processada, o conjunto de estados indicado pelo ponteiro no estado sendo analisado é percorrido, sendo adicionados ao conjunto de estados atual S_i todos os estados em que o símbolo à direita do ponto seja o símbolo não terminal do lado esquerdo da produção do estado em que a operação *completer* foi aplicada, sendo o ponto movido para a direita, indicando que o símbolo anterior foi processado. A gramática do estado em que a operação foi aplicada é mantida nos novos estados. O seguinte estado é então adicionado ao conjunto S_2:

$$A^1 \rightarrow aA^1\{Ad1(A^1, C^1, a, c)\} \quad b\ 0\ G^{3.2} \tag{14}$$

Os estados 12 e 13 são descartados pela operação *scanner*.

A operação *completer* é aplicada no estado 14, adicionando o novo estado:

$$K \rightarrow A^1.B^1C^1D^1 \quad \dashv\ 0\ G^{3.2} \tag{15}$$

A operação *predictor* é aplicada na sequência, adicionado os seguintes estados a S_2:

$$B^1 \rightarrow .bB^1\{Ad1(B^1, D^1, b, d)\} \quad c\ 2\ G^{4.3} \tag{16}$$

$$B^1 \rightarrow .b\{Ad2(D^1, d)\} \quad c\ 2\ G^{4.4} \tag{17}$$

As três operações continuam sendo aplicadas estado a estado, sendo os conjuntos de estados de cada passo do algoritmo apresentados resumidamente na sequência:

Passo $i = 3$, conjunto de estados S_3 e $X_4 = b$

$$B^1 \rightarrow b.B^1\{Ad1(B^1, D^1, b, d)\} \quad c\ 2\ G^{4.3} \tag{18}$$

$$B^1 \rightarrow b\{Ad2(D^1, d)\}. \quad c\ 2\ G^{4.4} \tag{19}$$

$$B^1 \rightarrow .bB^1\{Ad1(B^1, D^2, b, d)\} \quad c\ 3\ G^{5.1} \tag{20}$$

$$B^1 \rightarrow .b\{Ad2(D^2, d)\} \quad c\ 3\ G^{5.2} \tag{21}$$

Passo $i = 4$, conjunto de estados S_4 e $X_5 = b$

$$B^1 \rightarrow b.B^1\{Ad1(B^1, D^2, b, d)\} \quad c\ 3\ G^{5.1} \tag{22}$$

$$B^1 \rightarrow b\{Ad2(D^2, d)\}. \quad c\ 3\ G^{5.2} \tag{23}$$

$$B^1 \rightarrow .bB^1\{Ad1(B^1, D^3, b, d)\} \quad c\ 4\ G^{6.1} \tag{24}$$

$$B^1 \rightarrow .b\{Ad2(D^3, d)\} \quad c\ 4\ G^{6.2} \tag{25}$$

Passo $i = 5$, conjunto de estados S_5 e $X_6 = c$

$$B^1 \rightarrow b.B^1\{Ad1(B^1, D^3, b, d)\} \quad c\ 4\ G^{6.1} \tag{26}$$

$$B^1 \rightarrow b\{Ad2(D^3, d)\}. \quad c\ 4\ G^{6.2} \tag{27}$$

$$B^1 \rightarrow .bB^1\{Ad1(B^1, D^4, b, d)\} \quad c\ 5\ G^{7.1} \tag{28}$$

$$B^1 \rightarrow .b\{Ad2(D^4, d)\} \quad c\ 5\ G^{7.2} \tag{29}$$

$$B^1 \rightarrow bB^1\{Ad1(B^1, D^2, b, d)\}. \quad c\ 3\ G^{6.2} \tag{30}$$

$$B^1 \rightarrow bB^1\{Ad1(B^1, D^1, b, d)\}. \quad c\ 2\ G^{6.2} \tag{31}$$

$$K \rightarrow A^1B^1.C^1D^1 \quad \dashv\ 0\ G^{6.2} \tag{32}$$

$$C^1 \rightarrow .cC^2 \quad d\ 5\ G^{6.2} \tag{33}$$

Passo $i = 6$, conjunto de estados S_6 e $X_7 = c$

$$C^1 \rightarrow c.C^2 \quad d\ 5\ G^{6.2} \tag{34}$$

$$C^2 \rightarrow .c \quad d\ 6\ G^{6.2} \tag{35}$$

Passo $i = 7$, conjunto de estados S_7 e $X_8 = d$

$$C^2 \rightarrow c \quad .d\ 6\ G^{6.2} \tag{36}$$

$$C^1 \rightarrow cC^2. \quad d\ 5\ G^{6.2} \tag{37}$$

$$K \rightarrow A^1B^1C^1.D^1 \quad \dashv\ 0\ G^{6.2} \tag{38}$$

$$D^1 \rightarrow .dD^2 \quad \dashv\ 7\ G^{6.2} \tag{39}$$

Passo $i = 8$, conjunto de estados S_8 e $X_9 = d$

$$D^1 \rightarrow d.D^2 \quad \dashv\ 7\ G^{6.2} \tag{40}$$

$$D^2 \rightarrow .dD^3 \quad \dashv\ 8\ G^{6.2} \tag{41}$$

Passo $i = 9$, conjunto de estados S_9 e $X_{10} = d$

$$D^2 \rightarrow d.D^3 \quad \dashv\ 8\ G^{6.2} \tag{42}$$

$$D^3 \rightarrow .d \quad \dashv\ 9\ G^{6.2} \tag{43}$$

Passo $i = 10$*, conjunto de estados* S_{10} *e* $X_{11} = \dashv$

$$D^3 \rightarrow d. \quad \dashv \ 9 \ G^{6.2} \tag{44}$$

$$D^2 \rightarrow dD^3. \quad \dashv \ 8 \ G^{6.2} \tag{45}$$

$$D^1 \rightarrow dD^2. \quad \dashv \ 7 \ G^{6.2} \tag{46}$$

$$K \rightarrow A^1B^1C^1D^1. \quad \dashv \ 0 \ G^{6.2} \tag{47}$$

$$S \rightarrow K\{AdP(K)\}. \quad \dashv \ 0 \ G^{6.2} \tag{48}$$

$$\phi \rightarrow S. \dashv \quad \dashv \ 0 \ G^{6.2} \tag{49}$$

Ao aplicar a operação *scanner* no último estado não processado, estado 49, é adicionado ao conjunto de estados S_{11} o seguinte estado:

$$\phi \rightarrow S \dashv . \quad \dashv \ 0 \ G^{6.2} \tag{50}$$

Ao chegar ao fim do processamento de um conjunto de estados resultando em um estado S_{i+1} contendo apenas o estado 50, o algoritmo se encerra indicando reconhecimento da cadeia de entrada. A árvore de derivação pode ser montada percorrendo os estados em que foi aplicada a operação *completer*.

V. Considerações Finais

A implementação do algoritmo de Earley adaptativo apresentado no presente trabalho possibilitará a análise sintática de gramáticas adaptativas, sendo o primeiro componente que fará parte do novo arcabouço GrammarLab adaptativo.

O novo algoritmo traz como vantagens ser de propósito geral, podendo atuar em diferentes gramáticas sem que seja necessária uma implementação para cada uma, o que seria necessário caso fosse utilizado um autômato. A memória necessária para a execução da versão adaptativa do algoritmo de Earley será maior do que a utilizada em sua versão atual, uma vez que serão mantidas em memória diferentes gramáticas, que podem crescer exponencialmente de acordo com o tamanho da gramática inicial, de sua ambiguidade e da quantidade de ações adaptativas presentes na constituição. Além disso, na implementação atual do algoritmo no arcabouço GrammarLab, como forma de otimização de tempo de execução, a gramática é pré-processada, sendo gerados códigos fontes estáticos que devem ser compilados. Uma vez que a inclusão de adaptatividade requer um comportamento dinâmico, a estrutura atual do GrammarLab deve ser alterada, possivelmente representando um desafio para a implementação da abordagem proposta.

Como trabalhos futuros, o arcabouço continuará sendo evoluído de forma incremental, incluindo funcionalidades como a geração de classificadores baseados em gramáticas adaptativas, estimação de probabilidades de gramáticas adaptativas estocásticas e classificação de sequências e estruturas de RNAs contendo pseudonós.

Referências

[1] P. B. Menezes, Linguagens Formais e Autômatos: Volume 3 da Série Livros Didáticos Informática UFRGS. Bookman Editora, 2009.

[2] M. V. M. Ramos, J. J. Neto, and Í. S. Vega, Linguagens formais: teoria, modelagem e implementação. Bookman Editora, 2009.

[3] N. Chomsky, "On certain formal properties of grammars," Information and control, vol. 2, no. 2, pp. 137–167, 1959.

[4] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: A review," IEEE Transactions on pattern analysis and machine intelligence, vol. 22, no. 1, pp. 4–37, 2000.

[5] J. Earley, "An efficient context-free parsing algorithm," Communications of the ACM, vol. 13, no. 2, pp. 94–102, 1970.

[6] D. B. Searls, "The linguistics of DNA," American Scientist, vol. 80, no. 6, pp. 579–591, 1992.

[7] M. Brown and C. Wilson, "RNA pseudoknot modeling using intersections of stochastic context free grammars with applications to database search." in Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing, 1995, pp. 109–125.

[8] D. B. Searls, "Linguistic approaches to biological sequences," Computer applications in the biosciences: CABIOS, vol. 13, no. 4, pp. 333–344, 1997.

[9] E. Rivas and S. R. Eddy, "The language of RNA: a formal grammar that includes pseudoknots," Bioinformatics, vol. 16, no. 4, pp. 334–340, 2000.

[10] M. Sipser, Introduction to the Theory of Computation. Thomson Course Technology Boston, 2006, vol. 2.

[11] D. H. Younger, "Recognition and parsing of context-free languages in time n3," Information and control, vol. 10, no. 2, pp. 189–208, 1967.

[12] J. J. Neto, "Adaptive rule-driven devices-general formulation and case study," in International conference on implementation and application of automata. Springer, 2001, pp. 234–250.

[13] M. K. Iwai, "Um formalismo gramatical adaptativo para linguagens dependentes de contexto," Departamento de Computação e Sistemas Digitais (PCS)-Escola Politécnica, Tese de Doutorado, Escola Politécnica, Universidade de São Paulo (USP), São Paulo, SP (in portuguese), 2000.

[14] J. J. Neto, "Adaptive automata for context-dependent languages," ACM Sigplan Notices, vol. 29, no. 9, pp. 115–124, 1994.

[15] A. Machado-Lima, "Laboratório de geraçao de classificadores de seqüências," Master's thesis, Universidade de São Paulo, 2002.

Gilmar Pereira dos Santos possui graduação em Sistemas de Informação pela Universidade de Mogi das Cruzes (2006), especialização em Engenharia de Software pela Faculdade Impacta de Tecnologia (2012). Atualmente é aluno de mestrado do Programa de Pós-Graduação em Sistemas de Informação da Universidade de São Paulo, atuando nos seguintes temas de pesquisa: linguagens formais, reconhecimento de padrões, classificação funcional de RNAs, análise de sequências, aplicação de métodos adaptativos em problemas de bioinformática.

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 S ão 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.

Ariane Machado Lima possui graduação em Ciência da Computação pela Universidade de São Paulo (1998), mestrado em Ciências da Computação pela Universidade de São Paulo (2002) e doutorado em Bioinformatica pela Universidade de São Paulo (2006), pós-doutorado pela Universidade de São Paulo. Atualmente é Professora do Curso de Sistemas de Informação da Universidade de São Paulo e orientadora nos seguintes programas de Pós-Graduação: Sistemas de Informação e Interunidades em Bioinformática. Tem experiência na área de Ciência da Computação, com ênfase em Ciência da Computação, atuando principalmente nos seguintes temas: linguagens formais, reconhecimento de padrões, bioinformática, RNAs não codificantes, análise de sequências.