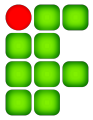

Instituto Federal do Ceará

DAVID PASSOS DE SOUSA
CARLOS DAVID LIRA
FRANCISCO DAVI CAMILO RIBEIRO
EDBERTO LIMA DE ANDRADE
FILIPE RODRIGUES DE MENDONÇA

Estudo de caso: Parsing sintático baseado em Autômatos Finitos Determinísticos

Fortaleza
5 de julho de 2017

DAVID PASSOS DE SOUSA
CARLOS DAVID LIRA
FRANCISCO DAVI CAMILO RIBEIRO
EDBERTO LIMA DE ANDRADE
FILIPE RODRIGUES DE MENDONÇA

Estudo de caso: Parsing sintático baseado em Autômatos Finitos Determinísticos

Trabalho com o objetivo de obter menção na disciplina de Aspectos Teóricos para o curso de Engenharia de Computação do IFCE.

Instituto Federal do Ceará

Orientador: Prof. Ernani Andrade Leite

Fortaleza
5 de julho de 2017

Sumário

Lista de Figuras

1 Introdução

Este trabalho fala sobre uma biblioteca, escrita em java, que foi desenvolvida para analisar gramáticas regulares. Para tanto, levou-se em consideração os conhecimentos adquiridos das disciplinas Aspectos Teóricos, Inteligência Artificial e no livro sobre Compiladores. Esta biblioteca serve como ponto de partida para outros trabalhos, como o de desenvolvimento de processadores de linguagem natural e compiladores. Os textos que descrevem como o trabalho foi realizado serão objetivos e enumerativos ao máximo, sem deixar de lado informações que sejam importantes para a reprodução deste, além disso, os exemplos serão extensivos e até prolixos, a finalidade é ter o maior numero de casos testados o possível, de tal forma que se possa comprovar a eficácia dos métodos utilizados.

Assuntos relacionados ao trabalho ou à sua caracterização serão enumerados até que se convirja para o assunto principal. A ideia é convergir para se situar, de modo que cada passo seguido possa ser interpretado isoladamente e se necessário modificado para melhor entendimento, desempenho ou generalização do problema.

Nos capítulos de 2 a 4 delimita-se o problema a ser tratado e suas aplicações, a solução proposta para sua resolução e os objetivos que norteiam o desenvolvimento do analisador sintático, além das dificuldades previstas pelas referências para o trabalho que se desenvolveu.

Nos capítulos de 5 a 8 os assuntos que servem de embasamento para o entendimento do que se fez serão discutidos, as ferramentas utilizadas enumeradas, as soluções propostas e métodos utilizados descritas, o conjunto treinamento utilizado delimitado e, por fim, apresentam-se os resultados e conclusões obtidos ao longo do desenvolvimento.

2 Problematização

A necessidade de escrever um analisador (*parser*) sintático surgiu juntamente com a vontade de se escrever um processador de linguagem natural (PLN). O analisador figura como um dos componentes mais importantes na construção de um PLN, pois ele é o elemento central que traduz a linguagem humana para uma interpretação computacional. O foco deste trabalho é, portanto, propor uma implementação do referido analisador.

É desafiante desenvolver um analisador sintático para a língua portuguesa, pois os aspectos semânticos e exceções encontradas na língua acabam por tornar sua caracterização de difícil modelagem. Além disso, a língua portuguesa é extensa e o analisador precisa ser escrito de maneira a abarcar o maior número de casos o possível, caso contrário, não se poderia garantir a confiabilidade dos programas baseados na biblioteca proposta.

A biblioteca desenvolvida propõe a implementação de uma gramática regular escrita sob os moldes da gramática normativa brasileira, proposta por Evanildo Bechara, apesar disto, ela não carrega consigo aspectos precisos a respeito da morfologia ou semântica, pois o objeto do trabalho ficaria demasiadamente extenso e o foco do desenvolvimento do analisador sintático seria perdido.

Em razão dos inúmeros problemas que existiram durante a fase de desenvolvimento do analisador, como sua caracterização extensa e por vezes ambígua, fez-se necessário utilizar alternativas para os códigos e regras gramaticais propostas, caso contrário não se teria um programa com solução precisa.

Por fim, a solução para modelar um problema tradicionalmente da língua portuguesa em um ambiente puramente computacional consistiu na recorrência às técnicas de modelagem por autômatos, gramáticas regulares e expressões regulares, todos estes, presentes em alguma fase de desenvolvimento do código que aqui propomos, mas nem sempre tão fáceis de se perceber, mais difíceis ainda quando tiveram que ser abstraídos para integrar o código da linguagem de programação utilizada.

3 Objetivos

- Criar o modelo diagramático do problema.
- Criar o modelo diagramático do módulo (*parser*).
- Implementar o problema computacionalmente.
- Melhorar a visualização do problema.
- Justificar o uso da ferramenta computacional.
- Justificar o uso dos aspectos teóricos para resolução do problema.

4 Dificuldades e eventualidades

Segundo o livro “Inteligência Artificial” (LUGER, 2013) os seguintes problemas poderiam ocorrer durante a implementação do código do módulo.

- O problema das escolhas que levam à sentenças não interpretadas.
- Dificuldade de controle da complexidade do problema.
- Diferenças entre a implementação da **geração** *versus* **derivação**.

5 Definições

5.1 A inteligência artificial

Não há uma definição clara do que seja inteligência, portanto também não poderíamos definir o que é inteligência artificial de maneira precisa, contudo, numa tentativa de tornar claro o que se deseja com esse trabalho é preciso que se dê uma definição formal para inteligência artificial que neste caso chamamos de “parte da ciência que trata de sistemas inteligentes, capazes de se adaptar a novas situações, raciocinar, compreender relações entre fatos, descobrir significados e reconhecer a verdade.”. (MICHAELIS, 2015).

Desta ciência diversas áreas de estudo são concebidas e dentre elas se destaca o estudo sobre processadores de linguagem natural, objeto de estudo deste trabalho.

5.2 Processadores de Linguagem Natural (PLNs)

Um do assuntos estudados pela Inteligência Artificial são os Processadores de linguagem natural. Estes não são um assunto novo, mas também não são um assunto de matéria já consolidada ou com pouco conteúdo. No livro “Inteligência Artificial” (LUGER, 2013) podemos perceber que para se criar um processador de linguagem natural com capacidades para interagir com pessoas um série de assuntos merecem ser estudados, lá eles estão classificados por níveis de análise para linguagem natural:

- A prosódia.
- A fonologia.
- A morfologia.
- **A sintaxe**.
- A semântica.
- A pragmática.
- O conhecimento do mundo.

Embora hajam diversos assuntos pertinentes à inteligência artificial, neste trabalho, vamos nos ater apenas ao estudo e implementação do **analisador sintático** (*parser* sintático), portanto, não se faz uma necessidade a definição de cada um dos níveis de análise dos PLNs, mas apenas este.

5.3 A sintaxe e o analisador sintático

> "A Sintaxe estuda as regras para combinar palavras e sentenças válidas e o uso dessas regras para analisar e gerar sentenças. Esse é o componente de análise linguística que foi mais bem sucedida.".(LUGER, 2013)

A representação computacional da sintaxe se da por meio de gramáticas regulares, a sintaxe de que este trabalho se ocupa esta relacionada àquela que trata da organização dos **lexemas** na estrutura do que se chama de **período** (sentença), ou seja, embora objetivo principal deste trabalho seja ser fiel à gramática normativa e à analise sintática que esta faz, a construção de uma gramática operada por um autômato vai mais além, permitindo que também possa ser usada de modo genérico para outras aplicações computacionais.

> "Regras de sintaxe são importantes não só em linguística, o estudo das linguagens naturais, mas também no estudo das linguagens de programação." (ROSEN, 2009).

Para tanto utilizou-se como base uma série de gramáticas e preceitos, de autores distintos, que integram a gramática proposta e implementada no módulo. Módulo este onde não delimita-se todos os casos linguísticos possíveis, pois o foco do trabalho seria perdido.

Em seu livro de "Inteligência Artificial" (LUGER, 2013), o autor, explica:

> "O primeiro estágio é a análise (parsing), que analisa a estrutura sintática das sentenças. Ela verifica se as sentenças são sintaticamente bem formadas e determina, também, uma estrutura linguística. Identificando as principais relações linguísticas como sujeito-verbo, verbo-objeto, substantivo-modificador, o analisador fornece um arcabouço para a interpretação semântica, que normalmente é representado por uma árvore sintática. O analisador (parser) da linguagem emprega conhecimento sobre a sintaxe da linguagem, a morfologia e um pouco da semântica.".(LUGER, 2013)

Durante a fase de desenvolvimento do analisador também se faz necessário entender o papel das gramáticas regulares, mais especificamente

das gramáticas dependentes (sensíveis) de contexto e de árvores de derivação, que constituem-se como ferramentas poderosas de modelagem de problemas envolvendo reconhecedores de linguagens.

5.4 Gramáticas Livres ou Sensíveis ao contexto

Existem diversas classificações de gramáticas e para este trabalho só é importante saber que se utilizou uma gramática sensível ao contexto, contudo o conhecimento de algumas informações podem ser úteis. Considere as informações a seguir:

> "Toda gramática regular é livre de contexto, mas nem todas as gramaticas livres de contexto são regulares."(WIKIPEDIA, 2016a).

Uma gramática é dita livre de contexto quando para cada estado S existe uma única regra de transição associada a partir de seu autômato correspondente e tal regra pode sempre ser utilizada. Já as gramáticas sensíveis ao contexto consideram a posição em que os elementos terminais ou não se enquadram na regra, veja um trecho encontrado em um fórum de grande credibilidade:

> "A context-sensitive grammar (CSG) is a grammar where each production has the form wAx $\rightarrow$ wyx, where w and x are strings of terminals and nonterminals and y is also a string of terminals. In other words, the productions give rules saying "if you see A in a given context, you may replace A by the string y."It's an unfortunate that these grammars are called "context-sensitive grammars"because it means that "context-free"and "context-sensitive"are not opposites, and it means that there are certain classes of grammars that arguably take a lot of contextual information into account but aren't formally considered to be context-sensitive.".(STACKOVERFLOW, 2017).

Entendido o que são gramáticas sensíveis ao contexto o resto do trabalho se torna de fácil compreensão e podemos continuar com a próxima ferramenta de modelagem importante, a árvore de derivação.

5.5 Árvore de derivação

A árvore de derivação se constitui como uma árvore que representa as transições realizadas pelo analisador. Nela pode-se observar claramente as regras de derivação utilizadas, a raiz e os nós terminais para uma determinada análise.

> "Uma derivação na linguagem gerada por uma gramática livre de contexto pode ser representada graficamente usando uma árvore ordenada enraizada, chamada de árvore de derivação ou *parse tree*." (ROSEN, 2009).

O interessante sobre essas árvores é que elas também podem ser modeladas por autômatos, de fato, o trabalho que se desenvolveu partiu de um autômato para se abstrair a busca em tal árvore. Essa abstração fica mais clara quando analisarmos o diagrama do autômato que foi desenvolvido nas páginas subsequentes. Por enquanto, é interesssante que se entenda também o que é o léxico.

5.6 O léxico

O léxico que compõe este trabalho foi proposto inicialmente para o programa "*grammar play*" (OTHERO, 2006) e foi adaptado para seu uso pelo java, pouco tempo mais tarde, quando se viu que a abordagem feita por ele em *prolog* seria melhor adaptada ao autômato que se desenvolvia, então, o léxico foi alterado compondo não mais agora regras declarativas impostas pela linguagem de programação *prolog*, mas uma implementação de um grafo escrito puramente em java.

Alguns itens lexicais serão enumeradas abaixo a fim de que se possua um embasamento sobre o que foi feito, contudo, não podemos neste trabalho enumerar cada item gramatical, pois o léxico deste módulo é extremamente extenso. Veja:

```
ama ,  Verbo
amam ,  Verbo
livros ,  Substantivo
meninas ,  Substantivo
amigo ,  Substantivo
os ,  Pronome
as ,  Pronome
incrível ,  Adjetivo
único ,  Adjetivo
insubstituível , Adjetivo
má ,  Adjetivo
da ,  P
```

```
eu ,  Pronome pessoal do caso reto
tu ,  Pronome pessoal do caso reto
elas ,  Pronome pessoal do caso reto
este ,  Pronome demonstrativo
esta ,  Pronome demonstrativo
algo ,  Pronome indefinido
nada ,  Pronome indefinido
como ,  Adjunto
professor ,  Substantivo
trabalho ,  Verbo
estudo ,  Verbo intransitivo
escreveu ,  Verbo transitivo
é ,  Verbo
português ,  Substantivo
às ,  Artigo
filho ,  Substantivo
compreende ,  Verbo
dos ,  Pronome
pais ,  Substantivo
esforço ,  Substantivo
redação ,  Substantivo
bela ,  Adjetivo
esgotado ,  Adjetivo
está ,  Verbo de ligação
manhã ,  Substantivo
prometia ,  Verbo transitivo
chuva ,  Substantivo
saíram ,  Verbo
de ,  Preposição
não ,  Advérbio
foram ,  Verbo
à ,  Preposição
festa ,  Substantivo
```

Definido o léxico e a gramática tem-se quase tudo que é necessário para criar um analisador sintático, resta-se apenas então criar a lógica que opera os lexemas (palavras) a partir das regras gramaticais, e é aí que os autômatos finitos entram.

5.7 Autômatos finitos (Máquinas de estados finitos)

Um autômato é uma representação finita de uma linguagem formal que pode ser um conjunto infinito. Autômatos são frequentemente classificados pela classe das Gramaticas regulares (também conhecida como Tipo 3 da Hierarquia de Chomsky), é uma restrição sobre a forma das produções, pode-se criar uma nova classe de gramáticas de grande importância no estudo dos compiladores por possuírem propriedades adequadas para a obtenção de reconhecedores simples.

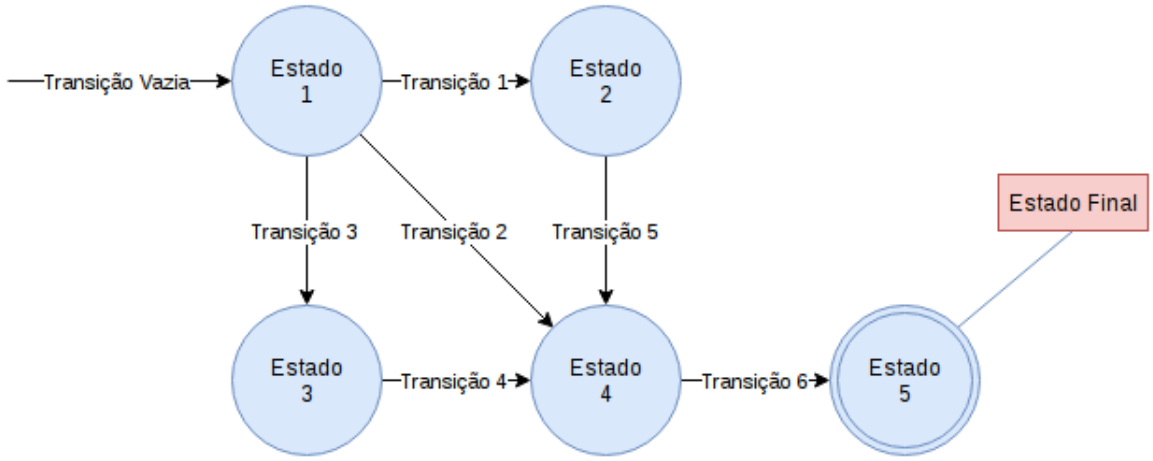

Figura 1 – Autômato Finito Definido

Ainda sobre autômatos, foi dito:

> "Na Ciência da computação teórica, teoria dos autômatos é o estudo dos objetos matemáticos chamados máquinas abstratas ou autômatos e os problemas computacionais que podem ser resolvidos usando esses objetos." (WIKIPEDIA, 2016b).

A justificativa para uso de autômatos se encontra no trecho a seguir:

> "(...) máquinas de estados finitos são a base para programas de verificação ortográfica, verificação gramatical, indexação ou busca em grandes textos, reconhecimento de fala, transformação de texto usando markup languages, como XML e HTML, e protocolos de redes que especificam como computadores se comunicam." (ROSEN, 2009)

Diagramas de estados:

> "Outra maneira de mostrar as ações de uma máquina é usar um grafo com arestas rotuladas, em que cada estado é representado por um circulo, as arestas representam as transições

> e estão rotuladas com a entrada e a saída para aquela transição.” (ROSEN, 2009)

5.8 Resumo definitivo

A biblioteca criada neste trabalho utiliza-se de **regras de reescrita** para especificar uma **gramática sensível ao contexto** simbolizada por um conjunto de **cadeias não terminais**. Definimos também um **dicionário** formado por lexemas (nós terminais), para os quais assumimos que existirão **transições vazias** que formam a **raiz** da **árvore de derivação** em nosso **autômato finito definido**.

Este emaranhado de conceitos que inicialmente parece complicado, fica mais claro quando são mostrados por meio de **diagramas de autômatos** e por representações de **gramáticas**. Nestas representações também fica claro o uso de **regras de derivação** para uso no **analisador sintático**, bem como para conhecimento da utilização de um modelo **top-down** em detrimento do **botton-up** e vice-versa.

Definido tudo o que se precisava pode-se então ter a certeza de que a decisão de seguir uma gramática mais normativa e formal, como a proposta por Evanildo Bechara, é uma alternativa viável. Mostrar-se-á a seguir como se fez tal analisador.

6 Ambiente de desenvolvimento

Mesmo simples, o projeto, foi desenvolvido ao longo de um tempo considerável e os softwares utilizados sofreram algumas modificações por seus fabricantes ao longo deste tempo, em sua versão mais atual o programa faz uso dos seguintes:

1. Netbeans IDE 8.2 (Livre)
2. JDK 8u92 (Livre)

6.1 Configuração do ambiente de desenvolvimento

Instalados todos os softwares acima listados é preciso configura-los.

6.1.1 JDK

O java development kit pode ser adicionado no caminho relativo a programas do windows ou linux (Path) em suas variáveis de ambiente, contudo esta opção é facultativa uma vez que a IDE pode ser configurada para reconhecer o caminho absoluto para suas bibliotecas. Lembre-se também de adicionar as bibliotecas de teste do jdk (Junit).

6.1.2 Netbeans

O netbeans foi escolhido por ser uma plataforma de desenvolvimento completa e bem recomendada pelos desenvolvedores da linguagem de programação java e depois de instalada basta que se configure-a para encontrar o JDK. Configuradas as bibliotecas faz-se necessário também que a biblioteca JUnit do java também se faça presente, este é um passo essencial e que costuma ser omitido, certificando-se que o JUnit foi configurado na IDE tudo ocorrerá bem.

6.1.3 O projeto em si

Na IDE em si basta que se crie um novo projeto e adicione os arquivos do projeto, desta forma, se o ambiente de testes do java estiver bem configurado o modulo será executado em modo de teste.

7 Implementação do Módulo

7.1 Lógica de funcionamento

No núcleo do módulo implementou-se um método que é o código para um autômato, com sua legibilidade e interpretação tal qual deveria ser. Um conjunto de entrada que representa um estado e pode seguir por meio de regras de transição para novos estados e continuar esta busca até se atingir uma meta definida pela gramática.

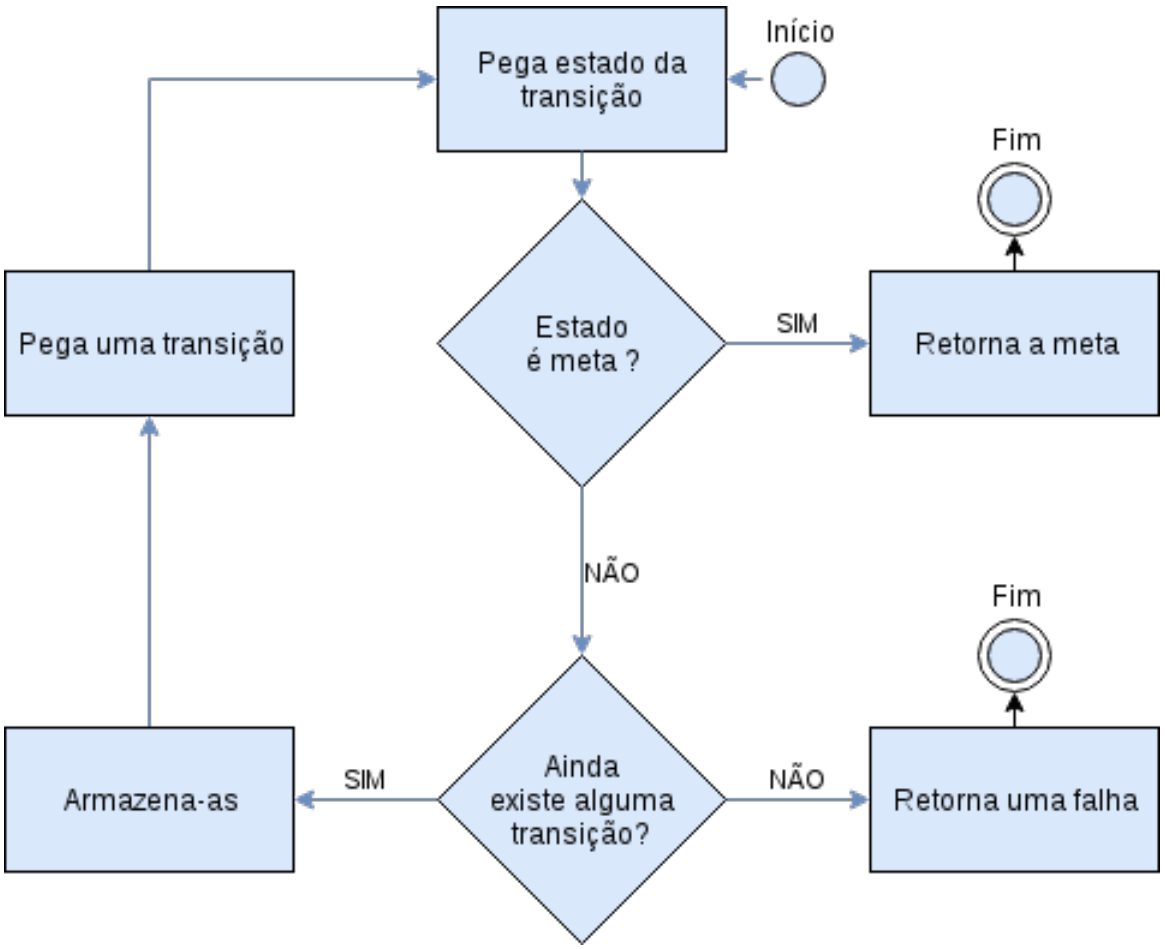

Figura 2 – Modelo diagramático do analisador sintático.

Faz-se necessário tornar clara a ideia de que a gramática **não é o autômato em si**, o autômato não passa de uma abstração que faz o uso da gramática regular para consumir as entradas.

O conjunto de entrada por sua vez é uma estrutura que relaciona um par de listas, do lado esquerdo o argumento de substituição e do lado direito a substituição em si.

7.2 Exemplo de uso

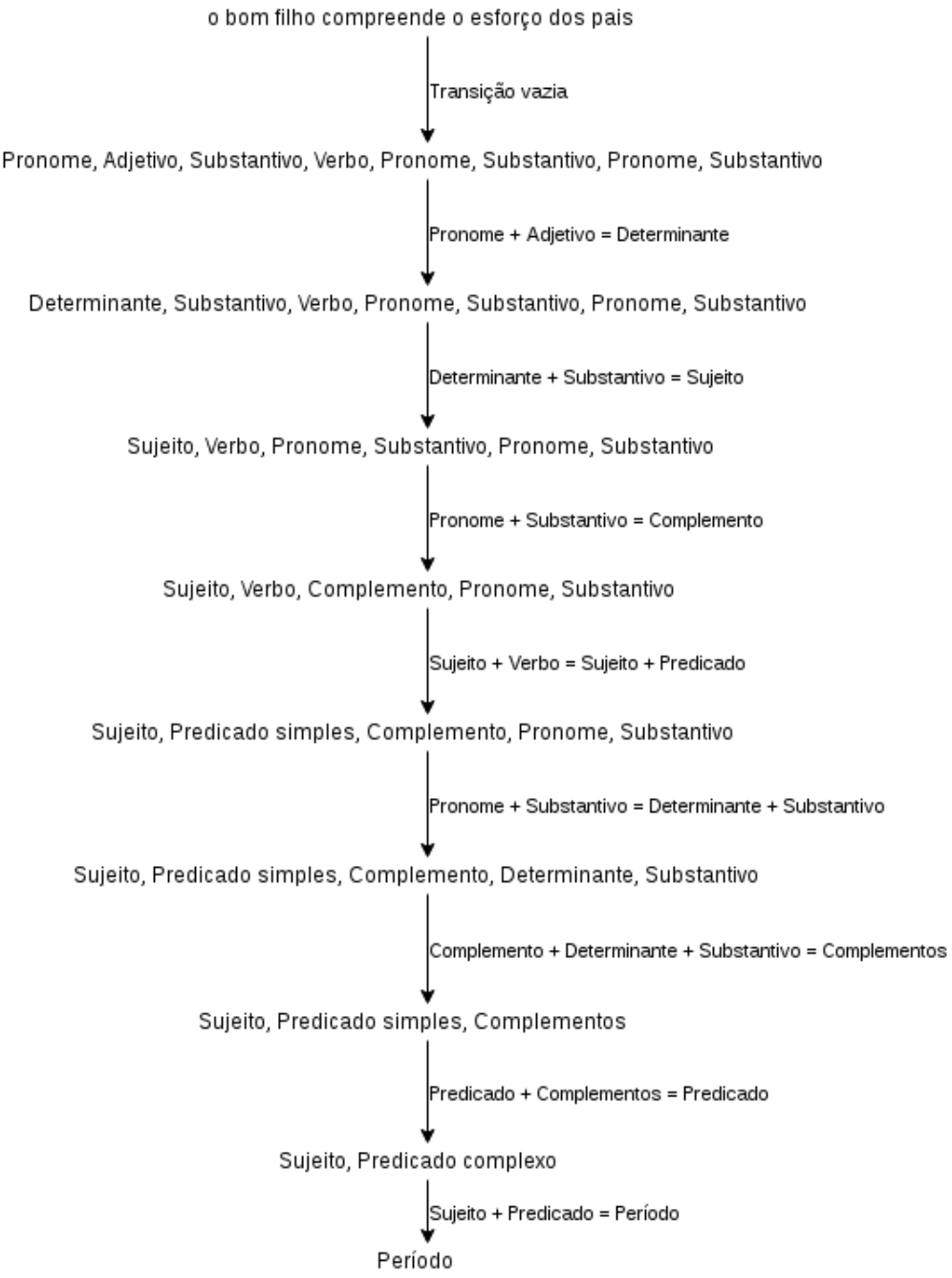

Figura 3 – Modelo diagramático do caminho percorrido pelo analisador.

A imagem acima trata de como o programa analisa a estrutura sintática do período (sentença).

7.3 “Conjunto treinamento”

Utilizou-se para fins de experimentação da eficácia do módulo os seguintes conjuntos de entrada:

```
(CASO FAVORÁVEL) eu estudo português às segundas-feiras
(CASO FAVORÁVEL) eu estudo português
(CASO FAVORÁVEL) eu estudo
(CASO FAVORÁVEL) estudo
(CASO FAVORÁVEL) os homens desejam paz
(CASO FAVORÁVEL) eu trabalho como professor
(CASO FAVORÁVEL) muitas crianças viram os pássaros
(CASO FAVORÁVEL) o bom filho compreende o esforço dos pais
(CASO FAVORÁVEL) joão escreveu uma bela redação
(CASO FAVORÁVEL) o livro está esgotado
(CASO FAVORÁVEL) esta manhã prometia chuva
(CASO FAVORÁVEL) todos os alunos saíram
(CASO FAVORÁVEL) alguns de nós não foram à festa

(CASO DESFAVORÁVEL) casa a é bonita
```

7.4 Justificativa do uso de AFDs

Inicialmente discutimos qual seria a melhor linguagem de programação para realizar a aplicação, de forma que fosse mais simples possível e que suas bibliotecas atendessem às necessidades de implementação da aplicação, inicialmente escolhemos o JAVA pois além da independência de plataforma possui extensa gama de bibliotecas que facilitam desenvolver este módulo.

A partir da definição da linguagem utilizada, para o desenvolvimento da aplicação, precisamos avaliar uma expressão de entrada como gramatical ou não e isto foi feito utilizando-se conceitos de gramáticas regulares.

Então dadas as facilidades inerentes à linguagem e a biblioteca “regex” já implementada por essa se justifica o uso de Java.

7.5 Problemas conhecidos

Regras gramaticais mal formadas podem resultar em uma recursão infinita, o que faz com a máquina virtual java gere uma exceção de estouro de pilha (*stackoverflow*). Para que isso não aconteça é necessário verificar se as regras de derivação geram estados convergentes e jamais cíclicos.

Pode-se inserir também, inadvertidamente, outro problema à biblioteca, caso não se atente ao que esta sendo inserido às regras lexicais ou gramaticais. Sempre leve em consideração que a biblioteca é *case sensitive*, isto é, diferencia minúsculas de maiúsculas, por isso, se o usuário não prestar atenção, o léxico ou a gramática podem acabar contendo regras não interpretáveis. Este erro é totalmente independente do programa e de difícil detecção.

7.6 Resultados

A implementação do módulo consistiu na criação de um método que se beneficiasse da ideia de consumo de estados de um autômato por meio de funções.

O diagrama do autômato que serviu de embasamento para a criação do núcleo do programa já foi mostrado, em correspondência à ele o código implementado ficou da seguinte forma:

```
public Lista<String> busca( Lista<String> tokens )
    {
        if( tokens.equals( meta ) )return meta;
        for( Lista<String> transicao : transicoes( tokens ) )
            if( busca( transicao ).equals( meta ) )
                return meta;
        return new Lista<>( "Não é a meta" );
    }
```

7.7 Gramática proposta

Segue um exemplo de construção da gramática proposta

```
Sujeito, Predicado simples
    -> Período
Verbo, Complemento
    -> Predicado simples
Predicado simples, Complemento
    -> Predicado complexo
Verbo, Substantivo, Determinante
    -> Verbo, Complemento
Verbo, Complemento, Complemento
    -> Verbo, Complementos
Verbo, Complementos, Complemento
    -> Verbo, Complementos
Verbo, Complemento, Complementos
    -> Verbo, Complementos
Verbo, Adjunto, Substantivo
    -> Verbo, Adjunto adnominal, Substantivo
Verbo, Adjunto adnominal, Substantivo
    -> Verbo, Complemento
Verbo, Adjunto adverbial, Substantivo
    -> Verbo, Complemento
Pronome, Substantivo, Verbo transitivo
    -> Sujeito, Verbo transitivo
Pronome, Substantivo, Verbo intransitivo
    -> Sujeito, Verbo intransitivo
Advérbio, Substantivo
    -> Determinante, Substantivo
Substantivo, Advérbio
    -> Substantivo, Determinante
Adjunto, Verbo
    -> Adjunto adverbial
Adjunto, Advérbio
    -> Adjunto adverbial
Adjunto, Adjetivo
    -> Adjunto adverbial
```

8 Limitações

Entre algumas limitações que podemos citar estão:

- Não abarca todas as regras de sintaxe que poderiam ser escritas.
- A gramática é dependente de contexto.
- Não foram implementadas regras de concordância.
- Não foram implementadas regras que tratem de pontuação.
- Não foram implementadas regras que tratem de capitalização.

Em geral o programa está limitado aos verbetes que conhece, seu “banco de dados”. O programa não é capaz aprender novos verbetes de uma forma "natural", pois este não se constituiu como foco de implementação, contudo o analisador que se descreveu pode ser atualizado, melhorado, embarcado, modificado para utilizações futuras.

9 Conclusão

Percebe-se então que o algoritmo proposto segue os modelos já consagrados para a criação de analisadores sintáticos, o código ficou simples, robusto e escalável, uma vez que diversas novas regras poderão ser criadas para melhora-lo sem que se precise alterar diretamente a lógica das instruções propostas. Além disso, o código é portável e utiliza bem os recursos computacionais e da linguagem sob a qual foi escrita, fazendo com que possa ser utilizado na maioria das circunstâncias.

Apêndice A - Lista de sinônimos

Léxico	Dicionário		
Analisador	*Parser*		
Gramática			
Item lexical	Palavra	*Token*	Lexema
Sentença	Período		

Referências Bibliográficas

CORDEIRO NILCEMARA LEAL MOLINA, V. F. D. Gisele do R. *Orientações e dicas práticas para trabalhos academicos*. Curitiba, PR: Editora Intersaberes, 2014.

LUGER, G. F. *Inteligência Artificial 6 Ed.* São Paulo, SP: Pearson Education do Brasil, 2013.

MICHAELIS. *Dicionário de Português Online*. 2015.

OTHERO, G. de Ávila. *Teoria X-Barra*: Descrição do português e aplicação computacional. São Paulo, SP: Editora Contexto, 2006.

ROSEN, K. H. *Matemática Discreta e suas Aplicações*. São Paulo, SP: AMGH Editora Ltda., 2009.

STACKOVERFLOW. *Context-free grammars versus context-sensitive grammars?* 2017. Disponível em: <https://stackoverflow.com/questions/8236422/context-free-grammars-versus-context-sensitive-grammars>.

WIKIPEDIA. *Enciclopédia Online*. 2016. Disponível em: <https://pt.wikipedia.org/wiki/Gram%C3%A1tica_livre_de_contexto>.

WIKIPEDIA. *Enciclopédia Online*. 2016. Disponível em: <https://pt.wikipedia.org/wiki/Teoria_dos_au%C3%B4matos>.