Flex, JFlex e GALS: Ferramentas de Apoio ao Ensino de Compiladores

Cinthyan Renata Sachs C. de Barbosa[1], Robson P. Bonidia[2], João Coelho Neto[3]

[1]Departamento de Computação – Universidade Estadual de Londrina (UEL)
Caixa Postal 10.001 - 86057-970 - Londrina - PR - Brasil

[2]Programa de Pós-Graduação em Bioinformática - Universidade Tecnológica Federal do Paraná (UTFPR) - 86300-000 - Cornélio Procópio - PR - Brasil

[3]Programa de Pós-Graduação em Ensino - Universidade Estadual do Norte do Paraná (UENP) - 86360-000 - Cornélio Procópio - PR - Brasil

cinthyan@uel.br, robservidor@gmail.com, joaocoelho@uenp.edu.br

Abstract. *This paper aims at analyzing educational tools for the compilers discipline, which is considered complex, once considered the number of techniques involved. Allied to this factor, there are not practically any comprehensive tools in this area. Focusing on this issue, the contribution of this research is a comparison between the Flex, JFlex, and GALS tools to the lexical analysis phase of the construction of a compiler. As final considerations, this research presented tools that can be used in the compilers teaching, providing the student with a wider interpretation at each stage of the compilation, making it possible for the Compiler discipline to be more attractive and didactic for students.*

Resumo. *Este artigo tem como objetivo analisar ferramentas educacionais para a disciplina de Compiladores, que é considerada complexa, dada à quantidade de técnicas envolvidas. Aliado a esse fator, praticamente inexistem ferramentas completas nesse tema. Visando esse problema, a contribuição desta pesquisa, é uma comparação entre as ferramentas Flex, JFlex e GALS para a fase de Análise Léxica da construção de um compilador. Como considerações finais, esta pesquisa apresentou ferramentas que podem ser utilizadas no ensino de compiladores, proporcionando ao aluno uma interpretação mais vasta em cada etapa da compilação, possibilitando que a disciplina de Compiladores possa ser mais atraente e didática para os discentes.*

1. Introdução

O Compilador é um dos módulos do *software* básico de um computador, cuja função é a de efetuar a tradução de textos redigidos em uma determinada linguagem de programação de alto nível para alguma outra forma que viabilize sua execução. Em geral, essa forma é a de linguagem de máquina, embora essa seja apenas uma das inúmeras possíveis alternativas viáveis [José Neto 1987]. Segundo Aho et al. (2008), o conhecimento sobre como organizar e escrever compiladores aumentou bastante desde que os primeiros começaram a surgir ao início dos anos cinquenta. Nessa época, esses programas foram considerados notoriamente difíceis de escrever. Ainda hoje, os princípios, técnicas e ferramentas discutidas em um curso de construção de compiladores são aplicáveis a uma gama de situações

que não necessariamente fazem parte de um projeto de um compilador [Debray 2002]. Muitas dessas são usadas em outras áreas da computação.

Infelizmente, um dos fatores que geram os elevados índices de evasão dos cursos na área de Computação, refere-se ao fato dos alunos apresentarem dificuldades nas disciplinas que abrangem os conteúdos de raciocínio lógico-abstrato, algoritmos e programação [Mcgettrick et al. 2005]. Alkimim e Mello (2010) chamam a atenção na formação superior em Computação, onde linguagens de programação e compiladores são conteúdos de relativa dificuldade de compreensão, portanto, torna-se necessário o desenvolvimento de recursos midiáticos que possam apoiar o ensino desses temas visando amenizar esses processos tão complexos. Ferramentas de cunho educacional voltadas à construção de compiladores são escassas, segundo Backes e Dahmer (2006), e poucas possuem capacidade suficiente para abarcar uma porção considerável das técnicas ensinadas em sala de aula.

A disciplina de Compiladores baseia-se em levar o aluno a perceber concretamente as diversas etapas envolvidas no processo de compilação, elaborando, ao final da disciplina, o projeto de um compilador funcional para uma gramática simples. Porém, muitos estudantes não têm a expectativa de vir a estar engajados em projetos de construção de compiladores em sua vida profissional e dessa forma, tendem a considerar o estudo de Compiladores, segundo Debray (2002) e Henry (2005), menos relevante que outras disciplinas, mesmo que as técnicas aprendidas possam ser utilizadas numa grande variedade de aplicações.

Uma primeira dificuldade, segundo Alkmim e Mello (2010), é a demora para que o aluno interprete a diferença entre manipulação de símbolos (características das máquinas reconhecedoras) e de valores, onde o desenvolvimento de programas é a atividade fim e o alcance de indicadores que apontam erros em etapas intermediárias da construção dos analisadores. Já uma segunda dificuldade, também apontada, é o alcance de indicadores que apontam erros em etapas intermediárias da construção dos analisadores. Por exemplo, na construção de uma Gramática Livre de Contexto (GLC) o aluno identifica erros ou problemas essencialmente após o desenvolvimento de todo o processo de construção dessa GLC, da implementação dos reconhecedores e do seu uso para validação de um código de teste.

A utilização de ferramentas educacionais voltadas para o ensino de Compiladores tem auxiliado o processo de ensino e de aprendizagem. Além disso, essas podem permitir, como enfatizam Backes e Dahmer (2006), uma melhor fixação dos conceitos e práticas apresentados durante o estudo de compiladores. As tecnologias aplicadas na educação englobam uma construção de saberes que parte da descoberta, da criação e do aprimoramento possibilitando ao aluno ter papel ativo, buscando desempenhar e resolver suas necessidades de uso. É necessário, salienta [Uliano 2016], que o professor perceba a importância de oportunizar e reconheça que a prática pedagógica mediada com as Tecnologias Digitais de Informação e Comunicação (TDICs) contribui muito em sala de aula trazendo novos processos de ensino e de aprendizagem. A utilização das TDICs no ambiente educacional é recomendada [Leite et al. 2013], pois essas tentam auxiliar na melhoria da qualidade do ensino em ambientes de aprendizagem mais ricos e motivadores para os discentes. Uma visão mais ampla de cada etapa da compilação se torna mais atrativa e didática, aliando teoria à prática.

Cooper e Torczon (2014) nos chamam a atenção que a maioria dos estudantes não se contenta em ler sobre essas ideias, onde muitas delas devem ser implementadas para que sejam apreciadas. Assim, o estudo da Construção de Compiladores é componente importante de um curso de Ciência da Computação. Sua prática é imprescindível, destacando a familiarização com ferramentas de geração automática de analisadores (Lex/Flex, Jflex/Jcup, etc), que permitem abordar as características reais de um compilador, na qual é conveniente apresentá-las para os discentes. Alkmim e Mello (2010) salientam que a ferramenta Flex é bastante explorada. O Flex (*Fast lexical analyzer generator*) e JFlex [JFlex 2018] são programas que recebem como entrada um arquivo com um conjunto de expressões regulares, as quais podem associar um *token* específico a cada uma delas. Também é possível recuperar o lexema de cada *token* reconhecido [Foleiss et al. 2009]. O GALS é um gerador de analisadores léxico e sintático [Furtado 2003], os quais são duas importantes fases do processo de compilação. Foi desenvolvido para ser uma ferramenta com propósitos didáticos, mas com características profissionais, podendo ser utilizada tanto no auxílio aos alunos da cadeira de Construção de Compiladores como possivelmente em outros projetos que necessitem processamento de linguagens [Gesser 2003].

Assim, o presente trabalho tem o objetivo apresentar e analisar as ferramentas Flex, JFlex e GALS que possibilitam auxiliar os professores na disciplina de Compiladores e proporcionar ao aluno uma interpretação mais ampla em cada etapa da compilação, tornando o processo mais atraente e didático. Um enfoque a fase de Análise Léxica será dado para que o aluno não desista já na primeira etapa do processo da construção de um compilador. Não serão descritas neste trabalho ferramentas Lex e Yacc [Levine et al. 1992], uma vez que tem bastantes trabalhos sobre essas, e nem tampouco ferramentas como SCC [Foleiss et al. 2009], C-Gen [Backes e Dahmer 2006] e CompilerSim [Costa et al. 2008] que já foram abordadas no trabalho de [Leite et al. 2013], por serem pouco conhecidas, porém em contrapartida essas nem sempre estavam disponíveis às academias universitárias. Além disso, a Fase de análise sintática também não será enfatizada aqui, pois envolve uma gama maior de fatores como a gramática descrita, técnicas de análises sintáticas e teriam que ser abordadas outras ferramentas de apoio como, por exemplo, o Bison [Levine 2009] que é um gerador de analisador sintático (assim como Yacc). Esse utiliza como seu gerador de analisador léxico padrão o Flex, que é uma versão aprimorada do Lex [Pinto 2015].

A abordagem metodológica é composta por uma revisão, por meio de livros e artigos na área de Compiladores, o qual visou compor o aporte teórico deste trabalho, bem como a análise das ferramentas supracitadas neste artigo.

Este trabalho está dividido em cinco seções: a primeira seção contextualiza a abordagem do estudo de compiladores; na segunda, a abordagem metodológica é definida; na terceira, o estudo das ferramentas para o ensino de Compiladores é realizado; na quarta, a análise das ferramentas; e na quinta e última seção, as considerações finais acerca da temática dos estudos serão apresentadas.

2. Compiladores

Com a introdução de linguagens de alto nível, surgiu a necessidade de programas *tradutores*, isto é, sistemas que traduzem programas (os chamados *programas fonte*) escritos em uma determinada linguagem para outra que viabilize sua execução, chamada de *pro-*

grama objeto. Por motivos históricos, os tradutores para linguagens de alto nível são chamados de *compiladores* [Kowaltowski 1993]. Como ressalta [Ricarte 2008], a grande motivação que catalisou o desenvolvimento de compiladores foi a demanda por uma maior eficiência de programação. Em termos técnicos, um compilador é um *software* que tem por finalidade traduzir ou converter um programa escrito em uma linguagem L_f para um *software* escrito em outra linguagem L_b, onde L_f seria a linguagem fonte e L_b linguagem alvo. Deste modo, o programa P_f nela escrito, é denominado programa fonte [Setzer e Melo 1988].

Desse modo, para que essa atividade de compilação ocorra é preciso passar por duas fases: a análise e a síntese. Sendo que a parte de *análise* divide o programa fonte nas partes constituintes e cria uma representação intermediária do mesmo. A *síntese* constrói o programa alvo desejado, a partir da representação intermediária [Aho et al. 2008]. A primeira fase da compilação é dividida em análise léxica, sintática e semântica.

Na análise léxica, enfoque deste trabalho, seu objetivo é separar a sequência de caracteres do texto de um programa-fonte em itens léxicos ou lexemas, que são sequências de caracteres com um significado coletivo. Os lexemas são classificados como identificadores, palavras-chave, operadores, constantes, símbolos de pontuação, entre outros [Aho et al. 2008]. Contudo, vale ressaltar que o analisador léxico realiza outras tarefas, como remover comentários e marcas de edição (tabulações, caracteres de avanço de linha e espaços), bem como relacionar o número de linha com possíveis mensagens de erro [Barbosa et al. 2009]. Logo em seguida, vem à análise sintática que lê os lexemas e valida a estrutura do programa criando a árvore sintática, por fim, a análise semântica é responsável por garantir as regras que são disparadas durante a análise sintática [Ricarte 2008].

3. Abordagem Metodológica

Esta seção visa apresentar os procedimentos efetivados nesta pesquisa. As etapas de pesquisa foram definidas como: **(1)** Levantamento bibliográfico do assunto, através de livros relacionados à temática de Compiladores; **(2)** Revisão de literatura em eventos específicos na área de Informática na Educação nos últimos 15 anos, a fim de averiguar quais ferramentas educacionais estão sendo utilizadas para auxiliar no processo de ensino e aprendizagem de Compiladores; **(3)** Processo de inclusão e exclusão dos artigos da Revisão de Literatura, por meio de etapas adaptadas de uma "Revisão Sistemática", utilizando apenas artigos envolvendo o conteúdo de Compiladores; **(4)** A escolha das ferramentas Flex, JFlex, GALS, por meio da análise de artigos relacionados às ferramentas de confecção de Compiladores; **(5)** Análise das ferramentas educacionais em uma turma da disciplina de Compiladores a nível de mestrado em Ciência da Computação, a qual foi feita por quatro participantes com Termo de Consentimento da pesquisa, sendo que esses analisaram os softwares comparando-os pelos seguintes quesitos: ambiente de trabalho, robustez, usabilidade e aplicação da fundamentação teórica. Esses quesitos foram classificados pelos autores, a fim de identificar articulações e procedimentos das ferramentas para o ensino de Compiladores. Feita a análise foram identificados possíveis quesitos sobre a temática a fim de contribuir para a área de Informática na Educação, principalmente para o ensino de Compiladores.

4. Estudos das Ferramentas para o ensino de Compiladores

Esta seção visa apresentar as ferramentas educacionais classificadas, voltadas para auxiliar no processo de ensino e aprendizagem de Compiladores.

4.1. Flex

A ferramenta Flex é uma versão livre da antiga ferramenta Lex. Flex significa "Analisador Léxico Veloz" (do inglês *Fast Lexical Analyzer*). Esse analisador é utilizado para reconhecer padrões léxicos em um texto. O Flex gera um código fonte usualmente na linguagem C, que é o analisador léxico [Levine 2009]. A especificação de entrada é realizada por expressões regulares e comandos em linguagem de programação. O analisador é gerado quando uma expressão é reconhecida, assim, os comandos associados com ela são executados [Gesser 2003]. A Figura 1 é um exemplo de como configurar o Flex. Vale ressaltar que esta ferramenta não tem interface, ou seja, o usuário deverá escrever o código em algum editor de texto. O código abaixo tem que ser adaptado de acordo com a necessidade.

```
%{
#include <stdio.h>
#include <math.h>
%}

Identificador  [a-zA-Z]*
Numeros  [0-9]
Espacos [\\t\n]

%%

for|Foward|foward|Function|function|If|if|Goto|goto|xor { printf( "Palvras Reservadas: %s\n", yytext);}
[.|,|;||,,|(|)|()|:|=|<|>|+|-|*|{|}|[|]] { printf( "Simbolos especiais: %s\n", yytext);}
[<>|<=|>=|:=|..|*,*] { printf( "Simbolos compostos: %s\n", yytext);}
{Identificador} { printf( "Identificador: %s\n", yytext);}
{Numeros} { printf( "Numero: %s\n", yytext);}
{Espacos} { printf( "Espacos: %s\n", yytext);}


%%
```

Figura 1. Configuração do Flex

Para compilar esse programa é preciso ter o $Flex$ instalado, assim como um compilador C qualquer. Assim, em uma linha de comando foi executado: *flex "nome do arquivo"*, isso irá criar um ficheiro *lex.yy.c* que deverá ser compilado na linha de comando. Por exemplo, com o compilador GNU, foi utilizado: *gcc lex.yy.c –lfl*. Esse comando gerou o analisador léxico, como pode ser observado na Figura 2.

```
if teste and teste2
Palvras Reservadas: if
 Identificador: teste
 Palvras Reservadas: and
 Identificador: teste
Numero: 2
Espacos:

then A:=1
Palvras Reservadas: then
 Identificador: A
Simbolos compostos: :=
Numero: 1
Espacos:
```

Figura 2. Analisador léxico em Flex - Exemplo em Pascal

É fácil encontrar uma documentação sobre esta ferramenta, para aprender como usá-la. Porém é preciso que o usuário esteja familiarizado com ferramentas de linha de comando.

4.2. JFlex

JFlex é um gerador de análise lexical (também conhecido como gerador de *scanner*) para Java, implementado também nessa linguagem. Foi desenvolvido por Gerwin Klein na Universidade de Princeton, com uma releitura da ferramenta Flex. O programa JFlex serve para criar uma classe Java que faz a análise léxica de qualquer arquivo texto. Esse faz o reconhecimento das linguagens por meio de tabelas de transição de autômatos determinísticos e de expressões regulares [Rojas e Mata 2005] [JFlex 2018]. Como pode ser observado na Figura 3, o texto é dividido por seções %%. Tudo o que está escrito nessa área vai ser gerado pelo JFlex. A classe gerada lê as entradas especificadas por meio do conjunto de expressões regulares. Assim, é gerado o programa (analisador léxico) que irá quebrar os *Tokens*, que são as palavras chaves, comentários, operadores, etc [Klein 2010].

```
%%
%class Lexer
%type Token

A = [a-zA-Z_]
N = [0-9]
WHITE=[ \t\r\n]
%{
public String lexeme;
%}
%%
{WHITE} {/*Ignore*/}

/* Operadores Aritméticos */

( "+" | "-" | "*" | "/" | "div" | "mod" )    {lexeme = yytext(); return OPERADOR_ARITMETICO;}
```

Figura 3. Exemplo da classe Lexer, a qual define os Tokens para a análise.

A Figura 4 demonstra o código utilizado para gerar a interface do analisador léxico do JFlex.

```
switch (token) {

case PALAVRA:
     resultado = resultado + "<Palavra_Reservada>" + lexer.lexeme + "\n";
     break;
case OPERADOR_ARITMETICO:
     resultado = resultado + "<Operador_Aritmetico>" + lexer.lexeme + "\n";
     break;
case OPERADOR_LÓGICO:
     resultado = resultado + "<Operador_Logico>" + lexer.lexeme + "\n";
     break;
case OP_RELACIONAL:
     resultado = resultado + "<Operador_Relacional>" + lexer.lexeme + "\n";
     break;
case SIM_ESPECIAL:
     resultado = resultado + "<Caracter_Especial>" + lexer.lexeme + "\n";
     break;
case SIM_ESP_COMP:
     resultado = resultado + "<Caracter_Especial_Composto>" + lexer.lexeme + "\n";
     break;
case ERROR:
     resultado = resultado + "<Erro, simbolo não reconhecido> \n";
     break;
case ID:
     resultado = resultado + "<ID>" + lexer.lexeme + "\n";
     break;
case NUMERO:
     resultado = resultado + "<Numero>" + lexer.lexeme + "\n";
     break;
```

Figura 4. Código utilizado para gerar a interface do analisador léxico.

A Figura 5 mostra o funcionamento do Analisador Léxico confeccionado pelo software JFlex. Em geral, as expressões regulares geradas por esse, têm um desempenho muito bom sem otimizações especiais.

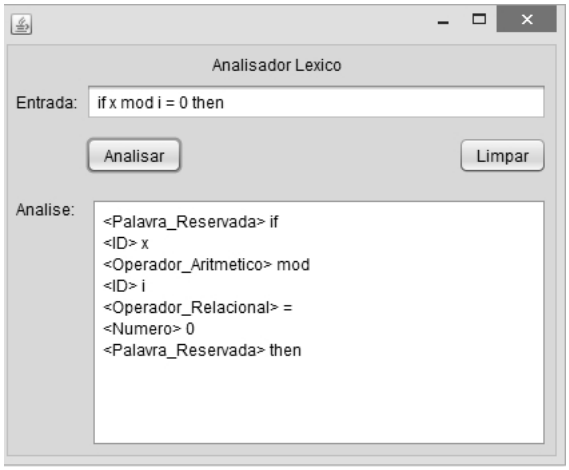

Figura 5. Analisador léxico - JFlex.

No entanto, a aprendizagem pode se tornar um pouco mais complicada para quem não tem nenhum conhecimento prévio da linguagem Java. Alguns dos objetivos do JFlex são: Geração de Scanner rápido e independência de plataforma. Contudo, essa ferramenta não gera um analisador sintático [Klein 2010].

4.3. GALS

GALS é um gerador de compiladores, que é utilizado para a geração automática de analisadores léxicos e sintáticos. Esse foi desenvolvido em Java, versão 1.4, podendo ser usado, segundo [Gesser 2003], em ambientes no qual haja uma máquina virtual Java. Esse gera os analisadores através de especificações léxicas baseadas em expressões regulares e sintáticas baseadas na GLC.

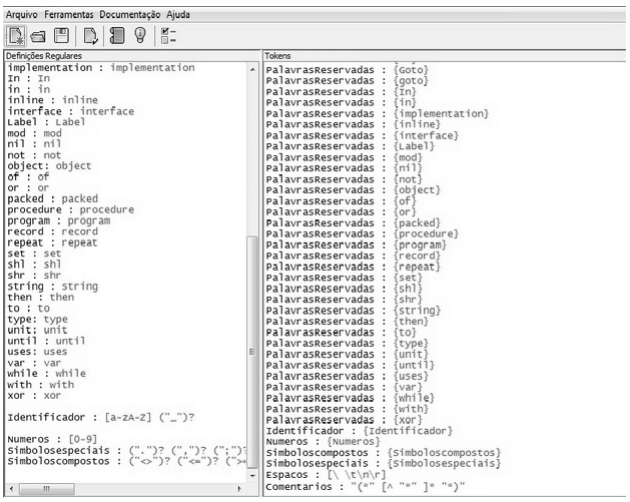

Figura 6. Configuração do GALS

É possível fazer analisadores léxicos e sintáticos independentes, bem como de maneira integrada [Gesser 2003]. Essa ferramenta oferece três opções de linguagens para a geração dos analisadores, que são: Java, C++ e Delphi. Segundo Gesser (2003) e GALS (2019), os aspectos léxicos de uma especificação no GALS são definidos pela declaração dos *tokens* e de definições regulares. Os *tokens* definem quais construções léxicas serão aceitas pelo scanner e o valor léxico de cada uma dessas construções. As definições regulares atuam como expressões auxiliares para definição de *tokens*, como pode ser observado na Figura 6.

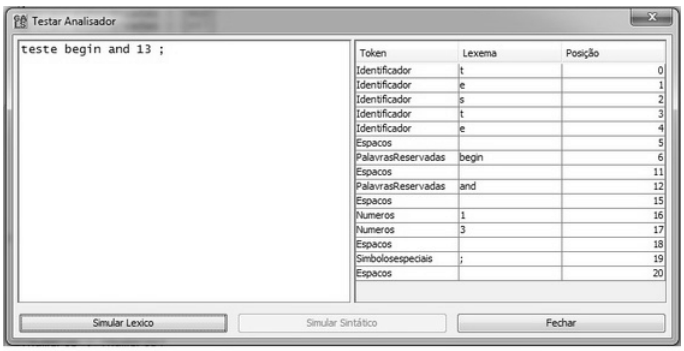

Figura 7. Analisador Léxico em GALS

O Analisador Léxico gerado utilizará as expressões regulares dos tokens para a criação de um Autômato Finito com múltiplos estados finais e dessa forma, cada um deles corresponde a um dos *tokens* declarados [GALS 2019]. A Figura 7 mostra um exemplo do analisador léxico criado pelo GALS em funcionamento. Desse modo, percebe-se que o GALS se apresenta como um ambiente didático e profissional [Gesser 2003].

4.4. Análise das Ferramentas

Os participantes analisaram os pontos fortes e fracos das ferramentas apresentadas, desse modo, classificações foram feitas, a fim de contribuir para um ambiente didático, obtendo-se o seguinte resultado separado pelos quesitos: Ambiente de Trabalho, Robustez, Usabilidade e Aplicação da Fundamentação Teórica. Além disso, a Tabela 1 apresenta resumidamente as características das ferramentas.

- **Ambiente de Trabalho:** Nesse quesito os sistemas Flex e JFlex não possuem um resultado eficaz, visto que a única interação com o usuário é feita por linha de comando, o que obriga o usuário a "programar" em um arquivo texto, trazendo incômodos para os aprendizes, porque estarão sujeitos a cometer muitos erros. Essa metodologia não é adequada para o aluno iniciante, porque há a necessidade de procurar erros no código fonte, tendo como agravantes o usuário não compreender o erro que cometeu. Contudo, a ferramenta GALS apresenta uma interface didática, que indica o erro no código, sendo que para utilização dele há apenas a necessidade de entender como funcionam suas expressões regulares.

- **Robustez:** Esse quesito é também chamado de *maturidade* [Brito Junior e Aguiar 2018], e as definições sugeridas é a capacidade do software de não apresentar falhas ou de não apresentar defeitos. O software Flex e JFlex são robustos, porém limitados, porque só podem ser utilizados para a confecção do analisador léxico.

Dessa forma, é necessário buscar outra ferramenta para terminar o processo de compilação. Outra limitação é gerar o analisador léxico apenas na linguagem C (no caso do Flex) e Java (para o Jflex). No entanto, GALS é uma exceção que gera analisadores léxicos, sintáticos e semânticos e tem a opção de gerar o analisador em três linguagens: Java, C++ e Delphi, flexibilizando assim o aprendizado do discente.

- **Usabilidade:** É também chamada por B. Junior e Aguiar (2018) de *qualidade de uso* e envolve vários critérios, como por exemplo, visibilidade, mapeamento entre o sistema e o mundo real, liberdade e controle ao usuário, consistência e padrões, prevenção e recuperação de erros, suporte ao usuário em reconhecimento e diagnóstico, flexibilidade e eficiência de uso, design estético e minimalista, ajuda e documentação, compatibilidade e adaptabilidade. As ferramentas Flex e JFlex utilizam como entrada um código fonte escrito em uma linguagem de programação. Desse modo, essa linguagem deverá ser ensinada ao discente, consumindo um tempo maior. Já o GALS necessita apenas que aprenda suas especificações léxicas, baseadas em expressões regulares e especificações sintáticas, baseadas em GLCs, que normalmente são bem lógicas e de fácil entendimento, reduzindo assim o tempo para que o aprendiz se atenha exclusivamente na aplicação da teoria absorvida na sala de aula.

- **Aplicação da Fundamentação Teórica:** Todos os softwares apresentados permitem a aplicação de parte dos fundamentos que engendram a disciplina de Compiladores. No entanto a ferramenta que mais se destaca é o GALS, por sua facilidade e simplicidade na construção de Compiladores, principalmente na confecção de Analisadores Léxicos.

Tabela 1. Comparativo das Ferramentas Flex, JFlex e GALS

	Flex	**JFlex**	**GALS**
Análise Léxica	**X**	**X**	**X**
Análise Sintática			**X**
Análise Semântica			**X**
Ambiente de Trabalho	**X**	**X**	**X**
Usabilidade			**X**
***Output* em várias linguagens**			**X**
Robustez	**X**	**X**	**X**
Simulação			**X**
Exibição das tabelas geradas			**X**
Material de Apoio	**X**	**X**	**X**
Aplicação Teórica	**X**	**X**	**X**
Análise de Erros			**X**

A partir desses resultados, percebe-se que o GALS apresenta um ambiente que pode ser propício para utilização educacional, fornecendo uma estrutura para gerar Analisadores Léxicos, Sintáticos e Semânticos, utilizando códigos em três linguagens de programação e evidenciando cada fase da compilação, além de realizar uma simulação

antes da geração do código para diagnóstico de um possível erro e exibir as tabelas geradas pelos analisadores. Contudo, há uma escassez de conteúdos de qualidade sobre o GALS e o JFlex, limitando o usuário a usar apenas a documentação da ferramenta.

No quesito material de apoio, o Flex possui vantagem sobre esse sendo maior e de melhor qualidade. Porém as ferramentas Flex e JFlex possuem um menor desempenho no quesito educacional, devido ao fato de consumir um maior tempo do aprendiz, necessitar de um conhecimento prévio em linguagem de programação, não apresentar interface dificultando encontrar um possível erro, realizar apenas a confecção do analisador léxico, o que limita o usuário e faz com que esse necessite buscar outra ferramenta para terminar o processo de compilação. Apesar disso, é correto afirmar que são ótimas ferramentas para confeccionar um Analisador Léxico profissionalmente, o que satisfaz a análise deste artigo.

5. Considerações Finais

Considera-se que, o objetivo deste trabalho foi alcançado, visto que as ferramentas apresentadas podem auxiliar e facilitar a compreensão do aluno nas fases da compilação. Consequentemente, depois de uma análise e comparação entre as ferramentas didáticas de compiladores, foi concluído que o Flex e JFlex não apresentam bons recursos para se trabalhar em sala de aula, devido ao consumo maior de tempo para aprender manuseá-los e o fato de não expor uma interface dificultando a procura de erros. Contudo são excelentes ferramentas para confeccionar analisadores léxicos.

A ferramenta GALS, ao contrário das outras analisadas, apresentou uma interface simples, com uma linguagem de fácil entendimento, além de possibilitar três linguagens de programação para que seja feita a geração de código e, principalmente, sendo possível visualizar todo o processo de análise léxica, sintática e semântica abstendo-se assim de procurar outras ferramentas para utilizar em sala de aula. Dessa maneira, demonstrando-se como uma ótima ferramenta didática.

Dessa forma, esta pesquisa apresentou ferramentas que podem ser utilizadas no ensino de Compiladores, proporcionando ao aluno uma interpretação mais vasta em cada etapa de compilação, possibilitando que a referida disciplina seja mais atraente e didática para os discentes. Espera-se que este artigo auxilie tanto discentes quanto docentes a explanar e adaptar grande parte do conteúdo na disciplina de Compiladores, bem como aponte a importância desse conhecimento possibilitando instigar e tornar mais tangível o uso dessas ferramentas nas diversas questões que envolvem o processo de aprendizagem de Compiladores.

Como trabalho futuro sugere-se ampliar este trabalho com as fases de análises sintática e semântica, e também verificar se o uso de ferramentas integradas [Oliveira 2014] melhoraria o processo de ensino e aprendizagem. Pretende-se ainda, fazer um trabalho didático para uma das mais úteis abstrações [Appel e Ginsburg 1998] usadas nos compiladores modernos na fase de análise léxica que são as expressões regulares.

Referências

Aho, A. V., Lam, M. S., Sethi, R. e Ullman, J. D. (2008). *Compiladores: princípios, técnicas e ferramentas, 2.ed.* São Paulo: Pearson Addison Wesley. 634p.

Alkmim, G. P. e Mello, B. A. (2010). Ferramenta de apoio às fases iniciais do ensino de linguagens formais e compiladores. In *XXI Simpósio Brasileiro de Informática na Educação*, João Pessoa, 1–4.

Appel, A. W. e Ginsburg, M. (1998). *Modern Compiler Implementation in C*. Cambridge: University of Cambridge. 544p.

Backes, J. e Dahmer, A. (2006). C-gen–ferramenta de apoio ao estudo de compiladores. In *XIV Workshop sobre Educação em Computação*, Porto Alegre, 1–8.

Barbosa, M. R., Silva, F. A., Victor, M. A., Feltrim, V. D., Mirisola, L. G., Gonçalves, P. C., Ramos, J. J. e Alves, L. T. (2009). Implementação de compilador e ambiente de programação icônica para a linguagem logo em um ambiente de robótica pedagógica de baixo custo. In *Anais do Simpósio Brasileiro de Informática na Educação*, 1–10.

Brito Junior, O. O. e Aguiar, Y. P. C. (2018). Taxonomia de critérios para avaliação de software educativo - tacase. In *XXIX Simpósio Brasileiro de Informática na Educação*, Fortaleza, 298–307.

Cooper, K. e Torczon, L. (2014). *Construindo Compiladores*. Rio de Janeiro: Elsevier. 656p.

Costa, K. A. P., Silva, L. A. e Brito, T. P. (2008). Auxílio no ensino de compiladores: Software simulador como ferramenta de apoio na área de compiladores. In *II Simpósio Internacional de Educação – Linguagens Educativas: Perspectivas Interdisciplinares na Atualidade*, Bauru.

Debray, S. (2002). Making compiler design relevant for students who will (most likely) never design a compiler. In *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education*, 341–345, New York, NY, USA. ACM.

Foleiss, J. H., Assunção, G. P., Cruz, E., Gonçalves, R. e Feltrin, V. (2009). Scc: Um compilador c como ferramenta de ensino de compiladores. In *Workshop sobre Educação em Arquitetura de Computadores*, São Paulo, 15–22.

Furtado, O. J. V. (2003). O ensino de linguagens formais vinculado ao ensino de compiladores. In *XI Workshop de Educação em Computação*, Campinas, 1–8.

GALS (2019). Gals home page. Disponível em: http://gals.sourceforge.net/. Acessado em 31/03/2019.

Gesser, C. E. (2003). *GALS-Gerador de Analisadores Léxicos e Sintáticos*. Florianópolis: Departamento de Informática e Estatística da UFSC. 150p.

Henry, T. R. (2005). Teaching compiler construction using a domain specific language. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*, 7–11, New York, NY, USA. ACM.

JFlex (2018). Jflex home page. Disponível em: http://jflex.de/. Acessado em 31/03/2019.

José Neto, J. (1987). *Introdução à Compilação*. Rio de Janeiro: LTC. 222p.

Klein, G. (2010). *Jflex user's manual*. Disponível em www.jflex.de/.

Kowaltowski, T. (1993). *Implementação de Linguagens de Programação*. Rio de Janeiro: Guanabara Dois. 189p.

Leite, V. M., de Barbosa, C. R. S. C., de Mattos Senefonte, H. C. e Neto, J. C. (2013). Uma seleção de compiladores educativos: Características e aplicações. *Encontro Regional de Computação e Sistemas de Informação, Manaus*, 1–4.

Levine, J. (2009). *Flex & Bison: Text Processing Tools*. O'Reilly Media, Inc.

Levine, J., Mason, T. e Brown, D. (1992). *Lex e Yacc*. O'Reilly Media, Inc., 2 edition.

Mcgettrick, A., Boyle, R., Ibbett, R., Lloyd, J., Lovegrove, G. e Mander, K. (2005). Grand challenges in computing: Education - a summary. *The Computer Journal*, 48(1):42–48.

Oliveira, H. B. (2014). *Um compilador, uma linguagem de programação e uma máquina virtual simples para o ensino de lógica de programação, compiladores e arquitetura de computadores*. Trabalho de conclusão de curso, Alfenas: Universidade Federal de Alfenas. 81p.

Pinto, T. T. S. (2015). *GGLL – Um gerador de analisadores sintáticos para gramáticas gráficas LL(1)*. Dissertação de Mestrado. Universidade de São Paulo. 103p.

Ricarte, I. (2008). *Introdução à compilação*. Rio de Janeiro: Elsevier. 264p.

Rojas, S. G. e Mata, M. A. M. (2005). *Java a Tope: Traductores Y Compiladores Con Lex/yacc, Jflex/cup Y Javacc*. Universidad de Málaga. 305p.

Setzer, V. W. e Melo, I. S. H. (1988). *A Construção de um Compilador*. Rio de Janeiro: Campus. 175p.

Uliano, K. C. M. (2016). *Tecnologia Digital de Informação e Comunicação (TDIC) na educação: Aplicativos e o mundo tecnológico no contexto escolar*. Monografia (Especialização) – Universidade Federal de Santa Catarina, Centro de Ciências da Educação. Curso de Especialização em Educação e Cultura Digital. Florianópolis, 50p.