CIFluxProgII: Ferramenta para auxiliar a avaliação de algoritmos utilizando Processamento de Linguagem Natural

Elisangela Maschio de Miranda[1], Anita Maria da Rocha Fernandes[1], Rudimar Luís Scaranto Dazzi[1], Rafael de Santiago[1]

[1] UNIVALI – Universidade do Vale do Itajaí
Centro de Ciências Tecnológicas da Terra e do Mar – CTTMar
Curso de Ciência da Computação / LIA – Laboratório de Inteligência Aplicada
Rua Uruguai, 458 – Centro / CEP: 88302-202 / Itajaí - SC

{elis,anita.fernandes,rudimar,santiago}@univali.br

Resumo. A comunicação é uma atividade diária do ser humano, sendo que a forma mais freqüente de comunicação é através da linguagem natural humana, ou seja, linguagem falada ou escrita. Tratando-se de ensino, a comunicação é um fator de grande relevância. Pensando-se por este ponto, desenvolveu-se uma ferramenta denominada CIFluxProgII, com o intuito de auxiliar o professor a avaliar algoritmos gerados por acadêmicos. Utilizando a teoria de compiladores e Processamento de Linguagem Natural, a ferramenta analisa o algoritmo do acadêmico e gera, através de um Sistema Especialista, a nota e comentário relevante ao problema em questão.

Palavras-chave: Processamento de Linguagem Natural, Algoritmos

Abstract. *The communication is a diary activitity of the human being, and the most frequent form of communication is through of human natural language, that is, spoken or written language. Dealing about teaching, the communication is a factor of great importance. Considering this point of view, a tool called CIFluxProgII was developed, with the aim of aiding the professor to evaluate algorithms generated by academic. Using the theory of Compilers and Processing of Natural Language, the tool analyses the academic's algorithm and generates, through a Expert System, the mark and important comment to the problem in question.*

Keywords: *Processing of Natural Language, Algorithms*

1. Introdução

Nos dias atuais a informação vem sendo disseminada de forma geral, e a informática tornou-se veículo importantíssimo neste processo. Por esta razão, as universidades oferecem a comunidade cursos vinculados a Informática e a Computação, com o objetivo de formar profissionais direcionados ao desenvolvimento tecnológico, e assim atender as necessidades da sociedade atual.

Ao ingressar em algum curso na área de computação o acadêmico terá contato com as disciplinas básicas do curso, e dentre elas destaca-se a programação. A introdução a programação, em grande parte dos cursos, é realizada através de uma disciplina denominada Algoritmos. Esta disciplina tem por objetivo ensinar o acadêmico a ordenar seu pensamento da forma em que trabalha uma linguagem de programação e aprender a desenvolver a lógica do programa em si.

A importância dessa disciplina será visualizada mais adiante, quando o acadêmico tiver contato com as linguagens de programação e precisar solucionar um determinado problema e passar esta solução em forma de comandos para o microcomputador.

O presente trabalho visa auxiliar o professor da disciplina de algoritmos na avaliação das soluções geradas pelos acadêmicos, com uma ferramenta que utiliza o conceito de processamento de linguagem natural e compiladores. Esta ferramenta efetua essa correção e ainda mantém o padrão e coerência na avaliação, bem como aproxima a forma de avaliação dos diversos professores da disciplina.

2. Processamento de Linguagem Natural

Barr e Feigenbaum (1986) colocam que o caminho mais comum para que as pessoas se comuniquem é falando ou escrevendo através de linguagem natural, seja em inglês, francês, português ou chinês. As linguagens de computadores possuem um formato mais rígido, para que possam ser convertidas em uma seqüência de instruções de computador. O estudo de processamento de linguagem natural procura fazer com que os computadores possam entender a linguagem natural humana, tornando-se mais fáceis de utilizar.

A principal preocupação da pesquisa sobre processamento de linguagem natural, conforme Oliveira (2003), é o uso da linguagem natural, ou seja, a utilização de agentes computacionais que se utilizam da linguagem natural para obter informações a respeito de outros agentes, sejam eles humanos ou máquinas, possibilitando ou causando mudanças em outros agentes, e, dessa forma, mudando o mundo.

Oliveira (2002) coloca que o Processamento de Linguagem Natural (PLN) está subdividida em duas sub-áreas: interpretação de linguagem natural e geração de linguagem natural.

Para que haja interpretação de uma sentença em linguagem natural, conforme Oliveira (1997), existe a necessidade de manter informações morfológicas, sintáticas e semânticas armazenadas em um dicionário, juntamente com palavras que o sistema compreenda.

O analisador morfológico, conforme informa Oliveira (2002), irá identificar expressões ou palavras isoladas em uma sentença, sendo auxiliado por delimitadores como pontuação e espaços em branco, e as palavras sendo classificadas conforme sua categoria gramatical. Neste contexto, a morfologia trabalha as palavras conforme sua estrutura, forma, flexão e classificação, no que se refere a cada um dos tipos de palavras.

Já a análise sintática, conforme Russell e Norving (2004), é o processo de construção de uma árvore de análise para uma cadeia de caracteres de entrada.

De acordo com Lacerda (1996), o termo "semântica" significa significado, ou pode também ser considerado como o estudo do significado, que em linguagem natural é a entidade ou ação que ela denota. A análise morfológica identifica as palavras individualmente, a análise sintática serve para determinar a estrutura de uma sentença, e a análise semântica serve para determinar o significado desta mesma sentença.

Lacerda (1996) destaca que pragmática é o estudo da comunicação e onde ela se situa no conjunto de necessidades de comunicação, emissores, receptores, tempos, lugares, ambiente, convenções lingüísticas e práticas culturais. Russell e Norving (2004) coloca que a análise pragmática leva em conta que palavras iguais podem possuir diferentes significados, dependendo do contexto em que está inserida. O significado não encontra-se nas palavras em si, mas na interpretação das palavras.

3. Compiladores

Conforme Aho e Seit (1995), um compilador é um programa que realiza a leitura de um outro programa escrito em uma linguagem fonte e o converte para um programa equivalente escrito em uma linguagem alvo. A compilação pode ser efetuada em duas partes: análise e síntese. Na análise o programa fonte é dividido nas partes constituintes e cria uma representação intermediária do mesmo. Já a síntese constrói o programa alvo desejado, partindo da representação intermediária, o que requer técnicas mais especializadas. As fases de um compilador são: analisador léxico, analisador sintático, analisador semântico, gerador de código intermediário, otimizador de código e gerador de código. As análises léxica, sintática e semântica foram as utilizadas no sistema, e as que serão descrita a seguir.

Dando continuidade, Aho e Seit (1995) informam que o analisador léxico faz a leitura dos caracteres, um por vez, transformando-os em tokens. Depois, o analisador sintático obtém a cadeia de *tokens* gerados na análise léxica e verifica se a mesma pode ser gerada pela gramática da linguagem-fonte. Ao tratar-se de análise semântica, a mais comum trata da verificação da consistência de tipos dos operandos envolvidos em operações aritméticas ou dos parâmetros passados a procedimentos.

4. Sistemas Especialistas

Cipriani (1997) coloca que especialistas são pessoas especializadas em resolver problemas específicos, sendo que esta competência é originada da experiência e conhecimento dos problemas que lidam.

Sistemas Especialistas, conforme Barreto (1997), conhecidos também por Sistemas Baseados em Conhecimento, são sistemas computacionais que procuram simular o comportamento de um especialista em um determinado domínio. Eles procuram imitar o processo humano de raciocínio, ao imitar o especialista humano.

Para a construção de um Sistema Especialista, Barreto (1997) informa que são necessários:

- o especialista, que é a fonte do conhecimento;
- transformar em dados e armazenar no computador o conhecimento a ser adquirido do especialista;

- gerar as regras de raciocínio, que são regras que mostram o raciocínio do especialista a respeito da resolução do problema;
- em grande parte dos casos existe a necessidade de um mecanismo que gere explicações de como o especialista chegou a determinada conclusão.

Quando trata-se de Sistema Especialista, pode-se classificar as regras de produção de duas formas:

- *Forward Chaining* (encadeamento para frente): partindo-se de um ponto inicial, chega-se a uma conclusão.
- *Backward Chaining* (encadeamento para trás): inicia com uma hipótese e procura valores para confirmar a mesma.

Fernandes (2002) coloca as seguintes vantagens dos Sistemas Especialistas:

- auxílio na redução de falhas humanas e aceleração de tarefas;
- flexibilidade, estabilidade e maior rapidez na resolução de problemas;
- aumento na qualidade e desempenho na resolução de problemas;
- combinação e preservação do conhecimento dos especialistas;
- integra várias ferramentas;
- não é afetado por questões psicológicas ou fatores externos;
- apresenta maior eficiência e otimização de resultados.

5. O Sistema Desenvolvido

O objetivo do trabalho foi construir um aplicativo para auxiliar professores da disciplina de Algoritmos, no curso de Ciência da Computação, na correção e avaliação de algoritmos gerados em Portugol. Para a correção dos algoritmos são utilizadas as análises léxica, sintática e semântica, referentes a teoria de compiladores, e através da técnica de Processamento de Linguagem Natural foi realizada a compreensão do algoritmo, usando-se a análise pragmática. Os erros encontrados durante as análises descritas acima foram revertidos em códigos e gravados em um arquivo texto separado. Estes códigos dos erros são lidos por um sistema especialista que classifica os tipos de erros, calcula a pontuação do acadêmico no algoritmo e realiza ponderações a respeito dos erros encontrados.

O CIFluxProgII utiliza como base o Editor/Compilador do CIFLuxProg. Ao iniciar este projeto, a ferramenta CIFluxProg contava com um ambiente gráfico onde o usuário digitava seu algoritmo em Português Estruturado, e um interpretador, responsável pelas análises morfológica (léxica) e sintática do algoritmo desenvolvido pelo acadêmico. Para desenvolver o CIFluxProgII, foram tomadas por base as funcionalidades existentes no CIFluxProg, e acrescentadas novas.

A. CIFluxProg

A ferramenta CIFluxProg – Construtor e Interpretador de Fluxograma para Programação - foi desenvolvida pelo GIA – Grupo de Inteligência Aplicada (Curso de Ciência da Computação / Centro de Ciências Tecnológicas da Terra e do Mar / Universidade do Vale do Itajaí) (Santiago e Dazzi, 2004).

Conforme Santiago e Dazzi (2004), o CIFluxProg permite a construção e execução de fluxogramas e algoritmos desenvolvidos em Português Estruturado (Portugol), o que torna o aprendizado dos conceitos de Algoritmos mais claros aos acadêmicos de primeiro período dos

cursos de Informática. Ao elaborarem o algoritmo e testarem no ambiente, os acadêmicos podem visualizar se o algoritmo em Portugol ou Fluxograma foi elaborado corretamente ou não, e se existem erros sintáticos e/ou semânticos.

Dando continuidade, Santiago e Dazzi (2004) informa que a ferramenta é composta por dois ambientes distintos: um ambiente para desenvolvimento de Fluxogramas, e outro ambiente para elaboração de algoritmos em Português Estruturado. Ambos contam um ambiente gráfico, arquivo de ajuda, e seu código pode ser executado através do interpretador de código. O usuário pode, também, executar funções padrão como abrir arquivos de código existentes e salvar algoritmos apresentados. Todas as soluções implementadas em um módulo podem ser abertas em outro, ou seja, se o acadêmico formulou um algoritmo em portugol, e deseja ver sua execução em fluxograma, é só abri-lo no módulo fluxograma.

O CIFluxProg, de acordo com Santiago e Dazzi (2004), suporta aninhamento. Aninhamento trata de se ter um símbolo dentro de outro símbolo, como por exemplo um laço de repetição dentro de outro laço de repetição. Este tipo de encadeamento deve ser suportado por um sistema que trabalhe com algoritmos, pois é bastante usado na resolução de problemas .

O usuário pode contar também com o teste de mesa, conforme o algoritmo gerado pelo mesmo for sendo executado, e com uma versão em portugol do algoritmo gerado em fluxograma. Conforme o acadêmico for inserindo os respectivos símbolos e dados, vai sendo exibido na caixa de texto na lateral direita a versão do mesmo algoritmo em Português Estruturado.

Santiago e Dazzi (2004) informam que a ferramenta foi levada à sala de aula para realização de testes. A ferramenta mostrou-se funcional, e um excelente auxílio ao professor da disciplina de Algoritmos, pelo fato de os acadêmicos poderem visualizar suas resoluções aos problemas apresentados, aumentando assim sua compreensão da matéria.

B. CIFluxProgII

As etapas metodológicas seguidas para a elaboração do trabalho foram:

- Estudo da Ferramenta CIFluxProg – foram realizadas reuniões com os desenvolvedores do CIFluxProg para conhecimento do sistema, e realizado testes para verificação das melhorias a serem realizadas.
- Levantamento dos quesitos a serem considerados quanto a correção e avaliação de algoritmos junto aos professores da disciplina de Algoritmos e Programação. Este levantamento foi realizado mediante reuniões com os professores da disciplina em questão, e análise de exercícios aplicados em acadêmicos da disciplina.
- Desenvolvimento das regras de produção referentes ao Sistema Especialista. Foi realizado um esboço inicial das regras de produção a serem desenvolvidas e mostradas na reunião com os professores da disciplina. Após esta reunião foram ajustados os valores e itens a serem avaliados.
- Análise da melhor forma de realizar a compreensão do algoritmo, ou seja, a análise pragmática.
- Implementação do protótipo.

Na figura 5.1 pode-se observar o modelo desenvolvido para construção do CIFluxProgII.

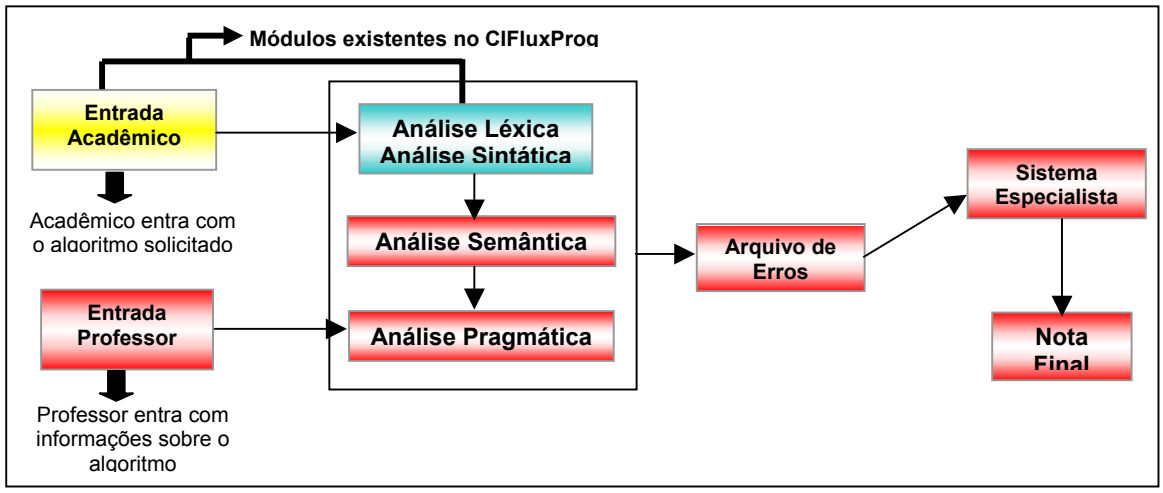

Figura 5.1: Módulos principais do projeto

O Sistema conta com dois módulos: o módulo do professor e o módulo do aluno.

No módulo do professor, é realizado o cadastro dos exercícios a serem desenvolvidos pelos acadêmicos, bem como informações relevantes a respeito do desenvolvimento do algoritmo, sendo que estas informações serão importantes para a realização da análise pragmática. Nesta primeira versão foram limitadas a 3 entradas fictícias e 3 saídas geradas por estas entradas. A tela do módulo do professor pode ser visualizada na Figura 5.2.

Figura 5.2: Tela do módulo do professor

Para a construção do módulo do aluno foram realizadas algumas alterações na ferramenta CIFluxProg original. Dentre elas pode-se citar a incorporação do módulo do Professor, o que ocasionou em mudanças na estrutura do módulo do aluno, e a alteração do código fonte existente, para que gerasse um arquivo texto com os códigos dos erros encontrados, para ser utilizado pelo Sistema Especialista (SE). Ao encontrar um erro durante as análises morfológica, sintática e semântica, o sistema informa o erro ao usuário para que o mesmo o conserte, antes de dar continuidade a análise do algoritmo. Os erros, além de serem mostrados ao usuário, são gravados em um arquivo texto separado, para serem utilizados pelo sistema especialista posteriormente. Quanto a análise pragmática, ela será realizada somente após as outras análises, e também, ao encontrar erro, o sistema grava-o no arquivo texto. Após todas as análises terem sido realizadas, o Sistema Especialista realiza a leitura dos erros encontrados e informa ao acadêmico a respectiva nota a respeito do algoritmo desenvolvido e realiza observações a respeito dos erros encontrados. A tela referente ao módulo do aluno pode ser visualizada na Figura 5.3.

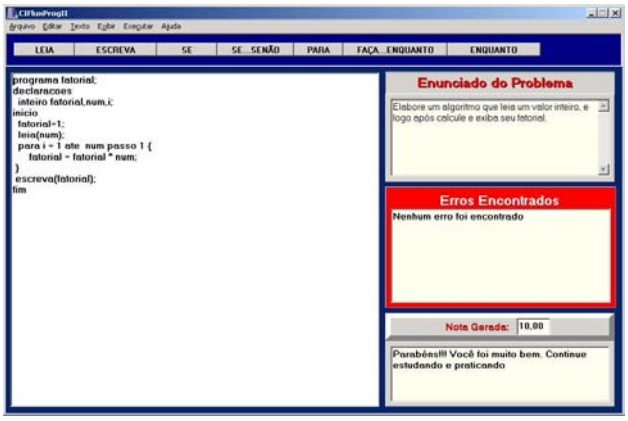

Figura 5.3: Tela do módulo do aluno

Para realizar a análise do algoritmo do acadêmico o CIFluxProgII realiza a análise léxica, sintática, semântica e pragmática do algoritmo, e através dos erros encontrados o Sistema Especialista gera a nota e comentários relevantes.

O CIFluxProgII foi baseado nos processos de análise léxica e sintática das ferramentas Lex e Yacc. Desta forma foi gerado o modelo sintático, semântico e a gramática do Portugol, o que criou funcionalidades que auxiliaram a interpretação do código.

Para realização da análise léxica os caracteres foram analisados da esquerda para a direita, sendo agrupadas as seqüências com significado coletivo e foram sendo determinadas as classes, o que é denominado *tokens*.

No CIFluxProg, através da aplicação de algoritmos sobre esta análise, buscou-se dados como identificação e criação de variáveis, contagem de linhas, armazenamento de constantes e identificação por tipo dos valores intrínsecos no portugol. Quanto às variáveis, o CIFluxProgII guarda seus respectivos nomes, valores e tipos em uma estrutura de dados, para que as mesmas possam ser utilizadas nas análises posteriores.

Já a análise sintática verifica se as palavras inseridas no código fonte estão gramaticalmente corretas. Tratando-se do CIFluxProgII, a análise sintática possibilitou uma maior preparação para a interpretação do algoritmo. Com os dados necessários identificados e estabelecidos, criou-se uma árvore de estrutura de dados para interpretação do algoritmo, que irá estabelecer a seqüência em que o algoritmo deve ser executado.

Uma pequena árvore é montada a cada estrutura analisada no portugol, contendo a seqüência e as informações necessárias para interpretação. Cada nó está diretamente relacionado com um determinado tipo de aplicação, como constantes, identificadores ou operadores. As constantes são armazenadas para serem atribuídas como valores necessários nas estruturas do portugol. A estrutura da árvore é processada por um algoritmo que realiza a interpretação, e após todas as operações que necessitem desta árvore, por outro que destrói sua estrutura.

Após a construção da árvore, que contém os passos para a interpretação, um algoritmo realiza a execução do código em portugol. Para que isso ocorra, realizou-se a identificação de todas as estruturas presentes no portugol – laços de repetição, desvios condicionais, operadores de atribuição, entradas, saídas – e, a cada passo da execução, foi-se analisando cada nó da árvore de interpretação e cada área que deve haver uma execução. O algoritmo aloca os recursos necessários e executa o fragmento de portugol.

A árvore de interpretação não é montada inteira. São montados pequenos pedaços, que são analisados, e logo após é montada outra parte, e assim sucessivamente, até o final do algoritmo.

A análise sintática e a análise semântica ocorrem em sincronia, conforme a árvore sintática vai sendo montada. O analisador semântico utiliza a árvore sintática montada e realiza tarefas como identificar operadores e operandos de expressões, reconhecer erros semânticos, realizar verificações de compatibilidade de tipo e analisar o escopo das variáveis.

Para a realização da análise semântica foi utilizada a tabela de símbolos criada durante a análise léxica, mas que foi sendo modificada enquanto da ocorrência das outras análises. Conforme a árvore sintática foi sendo montada, foi-se realizando pesquisas à tabela de símbolos para poder validar os tipos dos dados. Isso foi realizado atribuindo ações semânticas às produções da gramática, o que é conhecido como gramática de atributos.

Quando se fala em análise pragmática no desenvolvimento de algoritmos por alunos, trata-se de verificar se o que o acadêmico desenvolveu condiz com o solicitado pelo professor. No CIFluxProgII este processo foi realizado em duas etapas: a geração de entradas e saídas, e a procura por pontos chaves explicitados pelo professor.

Quando o professor cadastra o enunciado do algoritmo, ele cadastra informações importantes para a geração do algoritmo, tais como a quantidade de entradas e saídas, três possíveis entradas com as respectivas saídas, e informações referentes as estruturas necessárias para o desenvolvimento do algoritmo.

Na primeira etapa da análise é realizada a execução do algoritmo do acadêmico utilizando as três possíveis entradas, e são comparadas as saídas geradas pelo acadêmico com as saídas informadas pelo professor. Caso as saídas sejam iguais, a análise pragmática é considerada correta, sem erros, e o CIFluxProgII passa para a etapa do Sistema Especialista. Em caso de as saídas serem diferentes, o sistema passa para a segunda etapa. Nesta segunda etapa, são realizadas comparações com os dados cadastrados pelo professor, tais como se o algoritmo deve possuir laço de seleção, laço de repetição, qual a variação, se possui algum cálculo especial. Após essa verificação, o algoritmo passa para o Sistema Especialista.

Conforme são realizadas as análises e sendo encontrados erros, além de serem exibidas as mensagens de erro para o usuário também vão são gravadas em um arquivo texto. Quando são realizadas todas as análises, o sistema realiza a leitura do arquivo texto gerado, e realiza uma contagem dos erros de acordo com o código gravado. Através desta somatória é realizada a inferência do sistema especialista, que irá gerar a nota do acadêmico e um comentário a respeito do(s) erro(s) encontrado(s). Abaixo, segue uma das regras geradas como exemplo:

```
if (l01==0) && (l02==0) && (ST01==0) && (ST02==0) && (ST03==0) && (ST04==0) && (ST05==0) && (ST06==0) && (ST07==0) && (ST08==0) && (ST09==1) && (SM01==1) && (SM02==0) && (SM03==0) && (SM04==0) && (SM05==0) && (SM06==0) && (SM07==0) && (SM08==0) && (PRG01==0) && (PRG02==0) && (PRG03==0) && (PRG04==0) && (PRG05==0) && (PRG06==0) && (PRG07==0) {

printf("Nota: 9,00");

printf("Você esqueceu alguma palavra reservada, como também de declarar um identificador. Procure cuidar com estes erros no próximo algoritmo.");

}
```

Quando o sistema percorre o arquivo texto em busca dos códigos de erros encontrados, ele vai armazenando os erros em variáveis, para ter-se uma contagem de quantos erros por código foram encontrados no algoritmo. Foram criadas variáveis para acumular a contagem dos erros com nomes iguais aos códigos dos erros, e assim facilitar a montagem das regras de produção. Por exemplo, na regra mostrada acima, as variáveis ST09 e SM01 possuem o valor 1, ou seja, foram encontrados 1 erro de código ST09 e 1 erro de código SM01, sendo estes erros, respectivamente,

Palavra Reservada Faltante e Identificador não declarado. Ao encontrar estes dois erros o acadêmico perdeu um ponto, sendo sua nota 9,00. É escrito na tela a nota que o acadêmico tirou, e uma consideração a respeito dos erros encontrados.

As outras regras são formadas desta forma, sendo contados os erros e realizadas as inferências de acordo com a quantidade de erros encontrados.

Foram desenvolvidas 744 regras de produção, com encadeamento do tipo *forward chaining*. As regras foram estruturadas de forma a controlar a quantidade de erros por tipo, para assim poder gerar uma nota mais concisa.

6. Conclusões

A disciplina de Algoritmos e Programação é uma disciplina essencial no curso de Computação, e torna-se essencial que os professores avaliem os exercícios e provas de forma coerente e padronizada, e com este intuito foi elaborado o CIFluxProgII. Foram realizadas três tipos de avaliações para verificar a viabilidade do sistema: avaliação do sistema junto aos alunos, avaliação das funcionalidades junto aos professores e avaliação de eficiência.

A avaliação do sistema junto aos alunos realizou-se em duas etapas: em um primeiro momento foi apresentada a ferramenta aos alunos, e os mesmos desenvolveram dois algoritmos. Após o uso da ferramenta foi repassado um questionário aos acadêmicos de avaliação da ferramenta. Duas semanas após efetuada uma reavaliação da ferramenta, após a resolução dos problemas detectados e incorporação de novas melhorias. Foi aplicado o mesmo questionário, sendo assim possível verificar a melhoria do sistema e grau de satisfação dos acadêmicos. Verificou-se um crescente aumento de satisfação dos acadêmicos, tendo os mesmos compreendido melhor problemas clássicos de programação e sua lógica de solução. Isso se deu pelo fato dos acadêmicos poderem ter uma avaliação do algoritmo após seu desenvolvimento, o que possibilitou um desempenho melhor na solução dos erros cometidos.

Junto aos professores a avaliação deu-se em duas etapas: uma avaliação anterior ao desenvolvimento do sistema e outra após a conclusão do sistema. Na primeira etapa foi realizada reunião informal com os professores, onde foram levantados os pontos importantes na avaliação de algoritmos para desenvolvimento do sistema. Na segunda etapa os professores realizaram uma avaliação informal do sistema. Os professores mostraram-se satisfeitos com o sistema, mas solicitaram a verificação de uma maior quantidade de erros e que durante as análises o sistema não parasse para verificação de erros, o que será realizado em uma versão futura.

A avaliação de eficiência foi realizada por 5 acadêmicos e 2 professores, através de duas questões resolvidas pelos alunos e corrigidas pelos professores. O sistema mostrou-se eficiente quanto as análises léxica, sintática e semântica, mas necessita ser melhorada quanto a questão pragmática. Para modificações futuras sugere-se pensar na utilização de Redes Neurais Artificiais para análise pragmática. Outra sugestão seria a utilização de Processamento de Linguagem Natural para o sistema ir mantendo um diálogo com o acadêmico, e através de sugestões e questionamentos ir mostrando erros, resolução de problemas encontrados e sugestões.

As ferramentas Lex e Yacc mostraram-se satisfatórias, pois seu desenvolvimento foi realizado em menor tempo do que em linhas de programação e sua elaboração foi relativamente fácil.

A utilização de Sistemas Especialistas mostrou-se eficiente na primeira versão, mas em uma ampliação do sistema não seria mais. Foram desenvolvidas 744 regras de produção, que após a

aplicação de um filtro totalizaram 273 regras. A ampliação do universo de abrangência aumentaria consideravelmente o número de regras, tornando o sistema lento, menos eficiente e menos robusto. Para a realização de trabalhos futuros sugere-se a utilização de outras técnicas para geração de nota e parecer.

Como sugestão para trabalhos futuros recomenda-se a ampliação do sistema, inserindo vetores, matrizes, funções e ponteiros, o que seria de muita valia no ensino da disciplina.

7. Referências Bibliográficas

AHO, Alfred U.; SEIT, Ravi; ULLMAN, Jeffrey D. Compiladotes, princípios, técnicas e ferramentas. Editora LTC: Rio de Janeiro, 1995.

BARR, Avron; FEIGENBAUM, Edward A. The Handbook of Artificial Intelligence. Addison-Wesley Publishing Company, INC., 1986. vol. 1.

BARRETO, Jorge Muniz. Inteligência Artificial no limiar do século XXI. Florianópolis: J.M.Barreto, 1997.

CIPRIANI, João Aleixo. Protótipo de uma ferramenta para a construção de árvores genealógicas. Blumenau, 1997. Monografia (Graduação em Ciência da Computação) – Universidade Regional de Blumenau. Blumenau, novembro de 1997.

FERNANDES, Anita Maria da Rocha Fernandes. Inteligência Artificial. Disponível em:

[http://www.cttmar.univali.br/~anita/ia/d105/index.htm]. Acessado em: (05 de abril de 2002).

LACERDA, José Neves de. Generalização de fatos na compreensão de textos em linguagem natural**.** Florianópolis, 1996. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina. Florianópolis, novembro de 1996.

OLIVEIRA, Fabio Abreu Dias de. Processamento de linguagem natural: princípios básicos e a implementação de um analisador sintático de sentenças da língua portuguesa. Disponível em:

[http://www.inf.ufrgs.Br/procpar/disc/cmp135/trabs/992/Parser/parser.html]. Acessado em: (22 de abril de 2002 14:15]

OLIVEIRA, Itamar Leite de. Uma abordagem conexionista para resolução de anáforas pronominais**.** Florianópolis, 1997. Dissertação (Mestrado em Ciência da Computação) – Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina. Florianópolis, fevereiro de 1997.

OLIVEIRA, Alcione de Paiva. Processamento da Linguagem Natural. Disponível em:

[http://www.ufv.br/dpi/alcione/ia/ln.htm]. Acessado em: [22 de abril de 2003 21:35]

RUSSELL, Stuart; NORVING, Peter. Inteligência Artificial. Rio de Janeiro: Editora Campus, 2004. 2 ed.

SANTIAGO, Rafael de; DAZZI, Rudimar Luís Scaranto. Interpretador de Portugol. Artigo submetido e aprovado para o IV Congresso Brasileiro de Computação. Universidade do Vale do Itajaí – UNIVALI. Itajaí, 08 a 12 de outubro de 2004.