Atualização do Modelo do Aprendiz de Programação de Computadores com o Uso de Parser AST

Andres J. Porfirio[1,2], **Roberto Pereira**[1], **Eleandro Maschio**[2]

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)
Curitiba – PR – Brasil

[2]Coordenação do Curso de Tecnologia em Sistemas para Internet
Universidade Tecnológica Federal do Paraná (UTFPR)
Guarapuava – PR – Brasil

{ajporfirio,rpereira}@inf.ufpr.br, eleandro@hotmail.com

Abstract. *During their university education, Computer Science students have to develop computer programming skills. This specific topic represents challenges for both the student and the teacher. One of the main challenges is related to the monitoring of student progress by the teacher. Currently there are artifices that contribute to facilitate this monitoring, to mention: intelligent tutors systems and student profile modeling. These methods, however, still require intensive, and even manual, workload in order to feed the student model. This paper proposes a mechanism for feeding an apprentice model based on the use of an Abstract Syntax Tree parser and presents a comparison of the results of the same with respect to a human evaluator, the parser showed to be promising in the tests performed.*

Resumo. *O caminho a ser trilhado por alunos da área de Ciência da Computação possui como um de seus pilares a programação de computadores. Este tópico em específico remete a desafios tanto para o aluno quanto para o professor. Um dos principais desafios consiste no acompanhamento do progresso do aluno por parte do professor. Atualmente existem artifícios que contribuem para facilitar este acompanhamento, a citar: sistemas tutores inteligentes e métodos de modelagem do perfil do aprendiz. Estes métodos, por vez, ainda pecam no sentido de exigirem grande carga de trabalho para alimentação do modelo do aprendiz, muitas vezes feita de forma manual. Este trabalho propõe um mecanismo de alimentação para um modelo de aprendiz baseado no uso de um parser baseado em Árvores de Sintaxe Abstratas e apresenta um comparativo dos resultados do mesmo com relação a um avaliador humano, o parser se mostrou promissor nos testes realizados.*

1. Introdução

A programação de computadores é um dos fundamentos de todos os cursos da área de Ciência da Computação e, na maioria das vezes, apresenta sérios desafios tanto para o docente quanto para os alunos. Parte destes desafios se deve à dificuldade de mensurar o aprendizado e o progresso do aluno durante o processo. Por um lado o professor não possui formas precisas de avaliar os conhecimentos já adquiridos e, por outro lado, o

DOI: 10.5753/cbie.wcbie.2017.1121

aluno depara-se com a dificuldade em visualizar o próprio progresso em meio a um grande ecossistema de conceitos interligados e, muitas vezes, dependentes [Porfirio et al. 2016].

No ensino de programação de computadores, vem se tornando comum o uso de Sistemas Tutores Inteligentes (STI), que são ferramentas de apoio no processo ensino-aprendizagem. Tais ferramentas têm função de auxílio ao docente quanto à transmissão de conceitos, aplicação de exercícios, bem como a aplicação e consequente visualização do progresso dos alunos. Ademais, proporcionam aos alunos ambientes dinâmicos com funções que facilitam o estudo por conta própria (autônomo), fazendo papel do professor em ocasiões extra classe e, em alguns casos, facilitando a comunicação professor-aluno fora do ambiente de aula.

A literatura sugere benefícios na utilização de STI para o ensino de programação, tais como [Galasso and Moreira 2014], [Maschio 2013] e [Maschio and Direne 2015]. Cita-se o FARMA-Alg [Kutzke and Direne 2015], um ambiente de ensino, como parte do objeto de estudo deste trabalho. A principal característica deste sistema, que o faz pertinente para o presente trabalho, é a exigência de que os alunos desenvolvam o código de programação diretamente na ferramenta. Com isso, são armazenados registros de todas as ações tomadas pelo aluno ao longo da construção da solução.

Além disso, uma das funções interessantes no uso de STI é a capacidade de modelar o conhecimento e progresso do aluno ao longo do processo de ensino. Nesse sentido, vêm sendo utilizadas várias estratégias, na delimitação desta pesquisa cita-se: modelagem com o uso de Grafos Genéticos [Maschio and Direne 2015] e Redes Bayesianas Dinâmicas (RBD) [Vier et al. 2015].

Tanto o uso de Grafos Genéticos quanto RBD permitem que o professor atribua valores a determinadas características, representadas por nodos, que em conjunto são capazes de modelar o conhecimento do aluno. A inviabilidade se relaciona ao custo operacional de realizar esta tarefa para um grande número de estudantes. Agrega-se também a dificuldade e imprecisão em saber quanto um aluno aprendeu sobre um determinado assunto. Com base nisso, é levantada a hipótese de que uma forma automatizada de localizar indícios de aprendizado possa ser utilizada como ferramenta para valoração das características empregadas no mapeamento do conhecimento do aluno, trazendo como principal benefício o apoio ao professor, facilitando a identificação da evolução dos alunos.

1.1. Objetivos

O objetivo geral consiste na proposta de um método para atualização do perfil do estudante, baseada no uso de um parser baseado em Árvores de Sintaxe Abstratas (do inglês, *Abstract Syntax Tree* - AST). Este objetivo é alcançado por meio do desenvolvimento de uma extensão do FARMA-Alg capaz de analisar os códigos dos programas submetidos pelos alunos, em busca de indícios que levem à aquisição de perícias na área de programação de computadores.

São objetivos específicos: **(1)** Implementar uma rede bayesiana utilizando tecnologias compatíveis com o FARMA-Alg; **(2)** Implementar um mecanismo que faça uso de um parser AST para detecção de inícios de aquisição de perícia em determinados conceitos da área de programação de computadores; **(3)** Elaborar casos de teste para execução do parser; **(4)** Aplicar evidências na rede bayesiana com base nos resultados do algoritmo que emprega o parser; **(5)** Executar o método em casos de testes provenientes de situa-

ções reais de uso do FARMA-Alg; **(6)** Comparar os resultados do método proposto com os resultados obtidos a partir de uma análise realizada por humano profissional da área.

A sequência deste artigo é organizada conforme segue: a Seção 2 apresenta uma resenha sobre modelagem do perfil do aprendiz, a Seção 3 trata da atualização desse perfil. Na Seção 5, descreve-se o método proposto e em seguida, na Seção 6, são apresentados os resultados preliminares. Por fim, a Seção 7 traz as considerações finais.

2. Modelagem do Perfil do Aprendiz

Recentes trabalhos apontam a modelagem do perfil do aluno como artefato contribuinte para a melhoria da comunicação professor-aluno, melhoria na avaliação da aquisição de conhecimento do aluno e, consequentemente, benefícios no processo de ensino-aprendizagem.

O trabalho de [Maschio 2013] detalha a construção de um modelo de aprendiz para o contexto de programação de computadores. O modelo baseia-se na aquisição de princípios e desenvolvimento de perícias. A aquisição de princípios é descrita como a assimilação de conceitos e entendimento das regras que se aplicam ao objeto de estudo. O desenvolvimento de perícias, por sua vez, descreve a construção de conhecimento experiencial. A representação do modelo se deu por meio do uso de um grafo genético no qual as capacidades necessárias para o aluno se tornar um programador perito são representadas por nodos cujas interligações definem características como analogias, generalizações, especializações, pré-requisitos, entre outros. O autor cita resultados positivos na avaliação do método proposto, tais como melhor visualização do progresso do estudante e auxílio na escolha de enunciados de exercícios com base em perícias ainda não adquiridas.

Além disso, dentre as técnicas utilizadas na modelagem do conhecimento do aluno, encontra-se o trabalho de [Vier et al. 2015], que cita o uso de redes bayesianas. Segundo os autores, redes bayesianas são modelos probabilísticos que permitem lidar de forma rigorosa com a representação de conhecimentos em ambientes com muitas incertezas. Na proposta dos autores, a alimentação da rede (inserção de evidências) se deu a partir de resoluções de exercícios realizadas pelos alunos. Assim, se um aluno resolve corretamente um exercício sobre determinado assunto, então um nó correspondente da rede recebe um valor de 100%. O experimento foi avaliado com o uso de dados fictícios provenientes da aplicação de uma simulação citada como técnica de alunos virtuais.

Uma proposta de união das técnicas de grafo de perícias e redes bayesianas foi levantada por [Porfirio et al. 2016]. Os autores citam como benefícios desta união a capacidade de modelar o perfil do estudante norteando-se pela aquisição de perícias e a capacidade de expressar um ambiente de incertezas gerado por conceitos subjetivos a serem avaliados em um estudante de programação de computadores. Adicionalmente, desta união percebe-se que as conexões da rede representam dependências condicionais entre nodos, o que remete à estrita formalização dos pré-requisitos existentes na representação do conhecimento do aprendiz no contexto citado.

3. Alimentação do Perfil do Aprendiz

Tão importante quanto uma forma sólida e confiável de modelar o perfil do aprendiz é a maneira como são coletados os dados que o atualizam. A utilização de dados sintéticos

como a técnica de alunos virtuais citada por [Vier et al. 2015] é interessante para a análise do modelo. Entretanto, para adoção em ambiente real, faz-se necessário o uso de mecanismos que consigam identificar estas propriedades a partir do código-fonte desenvolvido pelo aprendiz, representando assim dados reais do estudante.

Dentre os métodos utilizados para detectar a evolução de conhecimento do aprendiz, encontra-se a técnica de questionários. O uso desta técnica como meio para acompanhamento da aprendizagem de programação é citado por [Pimentel et al. 2003]. Os autores descrevem uma metodologia que considera resultados de aplicações de questionários sobre conceitos de programação de computadores em turmas de cursos de graduação em Engenharia da Computação, Ciência da Computação e Sistemas de Informação.

Outra maneira bastante explorada consiste na elaboração de exercícios juntamente com seus casos de teste que, por sua vez, são compostos por entradas e saídas previamente conhecidas. Com base nisso, um sistema automático é capaz de executar o código do aluno, inserir dados de entrada e comparar o resultado final, julgando a solução como correta ou incorreta. Nesse sentido, [Galasso and Moreira 2014] fazem uso do sistema BOCA em conjunto com o Moodle, uma plataforma para ensino online. Com isso, os autores construíram um mecanismo para criação de questões no Moodle onde a resposta do aluno é um código de programação. O principal benefício citado foi a avaliação automática das respostas do aluno. Vale destacar que esse tipo de abordagem leva em consideração apenas o resultado final do algoritmo, e não considera os artefatos e técnicas que o aprendiz utiliza na solução.

Além de fazer uso de mecanismos que avaliam se solução do aluno está correta, também existem trabalhos trilhando um caminho contrário a essa premissa: verificando a presença de erros na solução submetida. [Porfirio et al. 2016] apontam diversos trabalhos cujo enfoque é a identificação de erros em códigos de programação, sendo que muitos desses trabalhos coletam informações fornecidas pelo compilador como mecanismos de apoio à correção do erro, a citar: mensagens de erro contendo sugestões de correção, adequação das mensagens de erro para melhor entendimento por parte de alunos iniciantes/inexperientes, tentativas de previsão de comportamentos e falhas do aluno, entre outros.

Conforme exposto, as pesquisas em sistemas tutores inteligentes focados em programação de computadores possuem várias vertentes quanto à modelagem do perfil do aprendiz. Entretanto, notam-se lacunas no que diz respeito ao uso de informações provenientes de análise do código fonte do aluno como base para alimentação deste modelo. Assim, este trabalho propõe a utilização de um parser AST como mecanismo para obtenção de parte destas informações.

4. Parser AST

Na área de ciência da computação, um parser possui função de receber uma entrada de dados e retornar subconjuntos de informações, organizando-as de acordo com algum critério. Especificamente tratando de programação de computadores, um parser recebe um código-fonte em linguagem de programação, interpreta-o e retorna elementos chamados tokens, que nada mais são do que detalhes de cada instrução que compõe o código do programa de computador.

Uma forma de retorno de informações de um parser se dá com o uso de AST.

Estas árvores representam a hierarquia de instruções do código-fonte e são construídas a partir de um processo que envolve: **(1)** pré-processamento do código-fonte, **(2)** análise léxica, **(3)** análise sintática e, por fim, **(4)** geração da árvore que representa a hierarquia de elementos que compõe o código-fonte [Cui et al. 2010].

Exemplifica-se a aplicação de um parser AST baseado no código-fonte expresso na Figura 1. O código é escrito em linguagem C, é possível perceber a presença de duas funções: *test* e *main*. Cada uma destas funções possui elementos internos, podendo ser: **(1)** declaração de variáveis e **(2)** chamadas de funções.

```
void test(){
  printf("test!");
}
int main(){
  int a = 2;
  test();
}
```

Figura 1. Exemplo de Código-Fonte

Uma vez executado o parser no código-fonte citado, é gerada uma AST na qual cada função do código C é armazenada em um nodo, cujos filhos são as instruções que a compõe. Além disso, cada nodo contém informações adicionais tais como os nomes e tipos literais de cada variável/função por exemplo. Aproveitando-se destas capacidades, propõe-se, na seção seguinte, um método para atualização do perfil do aprendiz com o uso destes recursos.

5. Método Proposto

O primeiro passo para avaliação do método é dado por meio da construção de um ambiente de testes, que consiste da conversão do grafo de perícias apresentado por [Maschio 2013] em uma rede bayesiana. A conversão tem como objetivo aproveitar-se dos benefícios citados por [Porfirio et al. 2016], tornando estrita a especificação de relações de dependência entre os conceitos representados pelos nodos, possibilitando a propagação de informações entre os nodos através de um mecanismo de inferência e, sobretudo, modelando um ambiente capaz de lidar com incertezas.

A rede foi implementada como uma página web dinâmica, capaz de receber parâmetros que alteram a aparência e característica da página apresentada. Então, o parser AST é aplicado a códigos-fonte de programas em linguagem C, os elementos da árvore resultante são utilizados na detecção de evidências para os nodos do rede.

A fim de testar a execução do algoritmo, considera-se a elaboração de casos de teste (códigos fonte) que contenham informação suficiente e relevante para a ativação (ou não) de determinados nodos da rede. Com isso, possibilitando a aplicação do método em uma fração dos nodos da rede, assim, mapeando parcialmente o conhecimento representado pelo modelo do aprendiz.

Reforça-se o estágio preliminar desta pesquisa, portanto os casos de teste são delimitados aos seguintes nodos da rede: **(1)** tipos e literais, **(2)** variáveis, **(3)** constantes e **(4)** variáveis x constantes. A delimitação tem como objetivo avaliar a hipótese proposta

e detectar possíveis limitações. Destaca-se ainda que, devido à natureza subjetiva das capacidades expressas em determinados nodos da rede, a técnica proposta não se aplica a todo o modelo do aprendiz, sendo necessário o uso de outras técnicas em conjunto com a proposta, ex: logs de compilação.

Por fim, cita-se a aplicação do método em casos de teste provenientes de informações reais obtidas da base de dados do FARMA-Alg. O objetivo dos testes é verificar a efetividade do método em códigos fonte que podem conter instruções inesperadas, lógica de programação diferente dos casos de teste sintéticos e, quaisquer artefatos provenientes da imprevisibilidade do raciocínio do aluno.

6. Resultados Preliminares

Dentre os resultados preliminares, têm-se até o momento: a implementação da rede bayesiana com sistema de evidências, o mecanismo de inferência, a alimentação automática de evidências na rede a partir de informações de um parser AST e os experimentos realizados.

6.1. Implementação do Grafo de Perícias sob a Forma de uma Rede Bayesiana

Com base no grafo de perícias descrito por [Maschio 2013], foi implementada uma rede bayesiana. A implementação foi realizada com o uso do *framework* Rails a fim de facilitar a integração com o STI FARMA-Alg, também baseado nesta tecnologia.

A construção do mecanismo que representa visualmente a rede também levou em consideração o carregamento de propriedades que são aplicadas em cada nodo. Assim, é possível adicionar, remover e alterar facilmente características de cada um deles. Dentre estas características, foram especificados parâmetros que representam as evidências que influenciam o estado de ativação do nodo. A ativação de um nodo pode depender de uma ou mais evidências (ponderadas por um percentual de influência).

A Figura 2 apresenta como exemplo o nodo "constantes", que diz respeito ao entendimento de conceitos de constantes. A ativação desse nodo leva em consideração três evidências: **(1)** o entendimento do conceito de tipos de literais, derivado do nodo "tipos de literais", com percentual de influência de 40%, **(2)** a capacidade de declaração e inicialização de uma constante, com percentual de influência de 30% e por fim, **(3)** a não tentativa de alteração do valor da constante ao longo do programa, com percentual de influência de 30%.

Cada nodo possui particularidades que o tornam único em relação às evidências que o ativam. No exemplo anterior, duas evidências (2 e 3) podem ser localizadas pelo parser a partir de uma análise dos tokens resultantes da criação da AST, e uma delas (a primeira) é resultado de uma inferência realizada pela rede bayesiana.

6.2. Mecanismo de Inferência

A rede bayesiana possui, como uma de suas principais características, o uso de mecanismos de inferência para ativação de nodos. O mecanismo permite que, dado um nodo X, com uma porcentagem de ativação Y, seja feita a propagação da informação Y para os nodos subsequentes de X, de tal modo que estes utilizem Y como parte do critério para sua própria ativação.

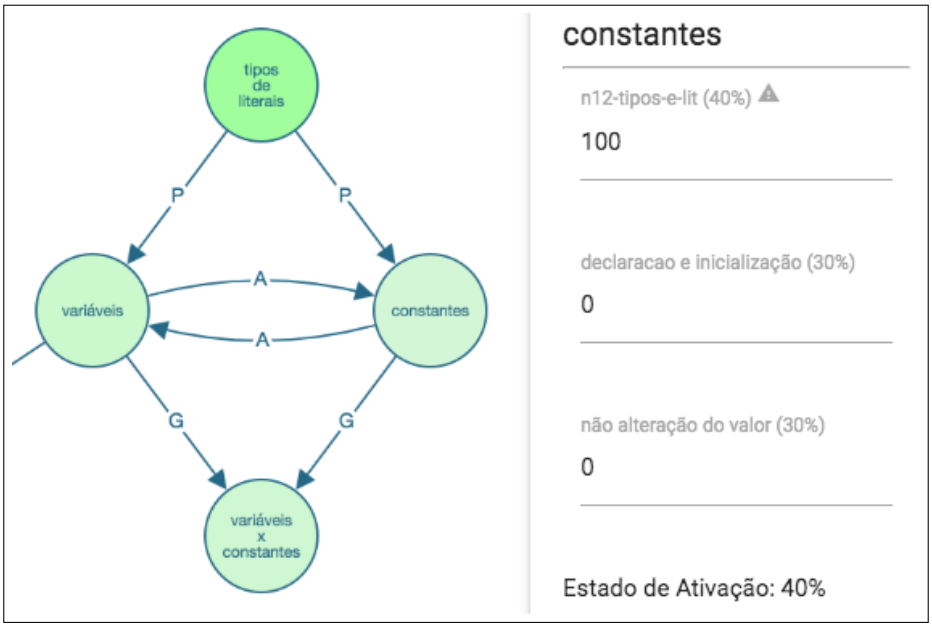

Figura 2. Evidências de um Nodo

Utilizando a situação hipotética representada na Figura 3, observa-se a inferência realizada no nodo "variáveis x constantes", que corresponde ao entendimento da diferença entre variáveis e constantes. Como o nodo "tipos de literais" possui 100% de ativação, o mecanismo de inferência fez propagação desta informação para o nodo "variáveis", que o usa como critério para ativação de 50%. Por fim, a porcentagem de ativação do nodo "variáveis" é propagada para o nodo "variáveis x constantes", resultado em 25% de ativação deste último nodo.

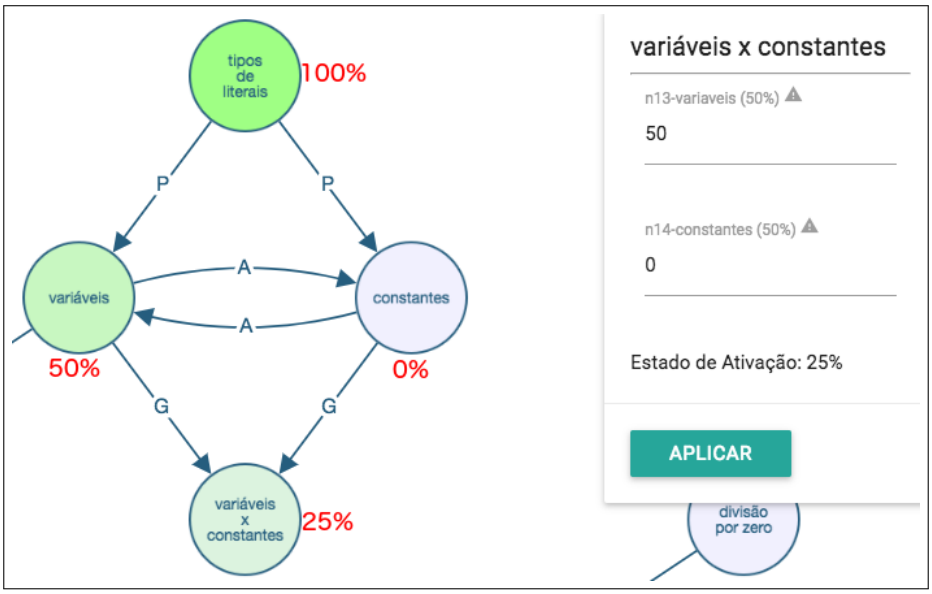

Figura 3. Mecanismo de Inferência

Assim, é possível perceber que os nodos podem fazer uso de várias fontes de informações para valorar as evidências que levam ao seu percentual de ativação. Uma das

formas citadas é através do uso de informações resultantes da aplicação de um parser. A seção seguinte detalha um exemplo deste caso.

6.3. Utilização do Parser AST

Conforme citado, um parser AST é capaz de segmentar e isolar informações a partir de códigos de programação. A aplicação de um parser desse tipo resulta em uma árvore de tokens que pode ser percorrida por algoritmos de programação. Ao longo do processo pode-se analisar as características de cada token e, diante do objetivo da pesquisa, detectar informações que possam ser utilizadas como evidências para ativação de nodos da rede bayesiana.

Exemplifica-se a aplicação do parser no código expresso na Figura 4. O código contém uma função principal (*main*) cujas instruções principais consistem na declaração de quatro elementos, uma variável e três constantes. Além disso, o código conta com duas expressões de atribuição.

```
int main(){
  int a = 2;
  const int b = 10;
  const int c = 23.5;
  const int d;
  a = 4;
  c = 6;
  return 0;
}
```

Figura 4. Código-Fonte Utilizado como Entrada do Parser

Retomando a situação exemplificada na Figura 2, onde foram detalhadas as evidências necessárias para ativação do nodo "constantes", é possível notar características existentes no código-fonte que permitem investigar se o aluno compreende ou não o conceito de constantes. Sob análise de um avaliador humano, a variável `a` e a constante `b` foram inicializadas corretamente, a constante `c` deveria ter sido inicializada com um valor inteiro, porém está incorreta ao utilizar ponto flutuante e, por fim, a constante `d` não foi inicializada em nenhum momento do código. Ademais, observando as operações de atribuição, nota-se que existe uma tentativa de alteração da constante `c`, o que indica uma falha no entendimento do conceito.

A execução do parser no código-fonte citado produz como resultado a árvore de sintaxe abstrata resumida na Tabela 1. A partir destes dados, é possível construir um algoritmo capaz de avaliar as duas evidências necessárias para ativação do nodo "constantes" da rede: **(1)** domínio de declaração e inicialização de constantes e, **(2)** domínio da premissa de constante que exige a não alteração do seu valor ao longo do código.

No exemplo específico citado, existem falhas graves no código: **(1)** a inicialização incorreta da constante `c`, **(2)** a não inicialização da constante `d`, e **(3)** a tentativa de alteração do valor da constante `c`. Todos estes indícios são detectáveis pelo parser conforme apresentado na Tabela 1, desta forma sugerindo viabilidade na utilização do mesmo com um dos mecanismos de inserção de evidências na rede bayesiana.

Tabela 1. Resultado da Aplicação do Parser

declaração	tipo	nome	tipo da inicialização ou atribuição	valor da inicialização ou atribuição
FunctionDef	Function	main		
Declaration	Int	a	IntLiteral	2
Declaration	Int (const)	b	IntLiteral	10
Declaration	Int (const)	c	FloatLiteral	23.5
Declaration	Int (const)	a		
ExpressionStatement	Assign	a	IntLiteral	4
ExpressionStatement	Assign	c	IntLiteral	6

6.4. Aplicação em Ambiente Real

Em adição aos testes descritos, foi elaborado um experimento para avaliar o método em condições reais. Para isto, o método foi aplicado em códigos-fonte provenientes da base de dados do FARMA-Alg.

A base de dados do FARMA-Alg é extensa e aumenta diariamente dado que a ferramenta se encontra em uso constante em disciplinas da área de programação dos cursos de graduação de várias universidades. Assim, foi selecionada uma fração da base de dados, composta por 29 programas, a partir dos seguintes critérios:

- Os códigos-fonte devem ser todos relacionados a resoluções de um único exercício, a fim de evitar grandes disparidades entre as soluções;
- A solução do exercício deve exigir o uso de variáveis;
- A linguagem de programação utilizada na solução deve ser a linguagem C;
- As soluções devem sempre ser extraídas de exercícios classificados como corretos pelo STI;
- Os códigos-fonte devem ser obtidos de duas turmas separadas, um arquivo por aluno, a fim de melhorar a diversidade de soluções.

Os seguintes indícios foram avaliados por um humano e, posteriormente, pelo parser AST: **(1)** capacidade de declaração de variáveis; e **(2)** capacidade de inicialização destas variáveis. No item **(2)**, foram levadas em consideração as seguintes características: **(a)** utilização de um tipo de dado compatível durante a inicialização e, **(b)** inicialização fazendo uso do resultado de uma expressão matemática. A primeira característica foi avaliada pelo algoritmo que usa o parser de modo validar se o tipo de dado atribuído é compatível, ou seja, se não existe perda de informação ou precisão. Já a última característica foi somente sinalizada pelo parser, não havendo a avaliação de compatibilidade entre o tipo de dado resultante da operação e o tipo de dado esperado pela variável.

A Figura 5 apresenta um gráfico que expressa um comparativo entre os resultados do avaliador humano e do parser levando em consideração a quantidade de declarações e inicializações encontradas. Os dados utilizados na confecção do gráfico foram extraídos da execução do método em 29 códigos-fonte de um exercício que exigiu, em média, o uso de 4 variáveis por solução.

É possível perceber que tanto o parser quanto o avaliador humano detectaram o mesmo número de declarações de variáveis (121). Também se pode notar que a identificação de variáveis inicializadas com expressões foi parecida, entretanto, o parser não

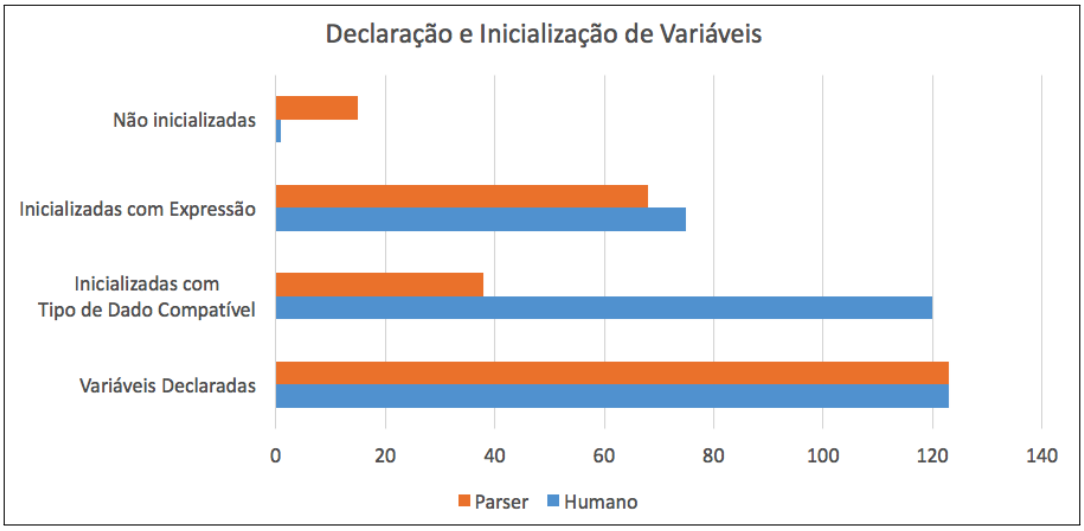

Figura 5. Comparativo de Detecção e Inicialização de Variáveis.

detectou 7 ocorrências que foram diagnosticadas pelo avaliador humano. A verificação do tipo de dado utilizado na inicialização mostrou-se uma deficiência do algoritmo que faz uso do parser, isto se deve ao fato de que inicializações com expressões não são completamente avaliadas pelo algoritmo (apenas sinalizadas). Além disso, cita-se o fato de que 14 variáveis foram marcadas incorretamente pelo parser como não inicializadas.

7. Considerações Finais

Diante do apresentado, destacou-se a capacidade do parser AST de detectar características de cada elemento presente no código-fonte. Com isso, foi apontada a possibilidade de utilização destas informações como parte do mecanismo de inserção de evidências na rede bayesiana, desta forma atuando como um mecanismo automático de atualização do modelo do aprendiz.

Cita-se como um benefício do método a capacidade de atuação em códigos-fonte independente das características e objetivos dos mesmos, permitindo que seja aplicado nos mais diversos contextos. Foram apresentados exemplos de códigos em linguagem C, contudo, qualquer linguagem capaz de ser interpretada por um parser AST pode ser contemplada com este mecanismo.

Por fim, devido ao estado de progresso da pesquisa, classificado como parcial, são citadas algumas limitações, tais como: aplicação apenas em um fragmento do conjunto de perícias representado pela rede e, incapacidade de avaliação de expressões matemáticas, sendo considerados apenas valores literais.

Referências

Cui, B., Li, J., Guo, T., Wang, J., and Ma, D. (2010). Code comparison system based on abstract syntax tree. In *2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT)*, pages 668–673.

Galasso, R. H. and Moreira, B. G. (2014). Integração do ambiente boca com o ambiente moodle para avaliação automática de algoritmos. *Anais do Computer on the Beach*, pages 22–31.

Kutzke, A. R. and Direne, A. I. (2015). *FARMA-ALG: An Application for Error Mediation in Computer Programming Skill Acquisition*, pages 690–693. Springer International Publishing, Cham.

Maschio, E. (2013). *Modelagem do Processo de Aquisição de Conhecimento Apoiado por Ambientes Inteligentes.* Tese de doutorado, Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná (UFPR).

Maschio, E. and Direne, A. (2015). Multiple external representations to support knowledge acquisition in computer programming. *Brazilian Journal of Computers in Education*, 23(03):81.

Pimentel, E. P., de França, V. F., Noronha, R. V., and Omar, N. (2003). Avaliação contínua da aprendizagem, das competências e habilidades em programação de computadores. In *Anais do Workshop de Informática na Escola*, volume 1, pages 533–544.

Porfirio, A., Maschio, E., and Direne, A. (2016). Modelagem genérica de aprendizes com Ênfase em erros na aquisição de habilidades em programação de computadores. *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*.

Vier, J., Gluz, J., and Jaques, P. (2015). Empregando redes bayesianas para modelar automaticamente o conhecimento dos alunos em lógica de programação. *Revista Brasileira de Informática na Educação*, 23(02):45.