Montanha de Chomsky: jogo tutor para auxílio no ensino de Teoria da Computação

Lucas Sampaio Leite[1], **Maria Aparecida A. Sibaldo**[2], **Tiago B. A. de Carvalho**[2], **Rodrigo de Souza**[1] *

[1]Departamento de Estatística e Informática – Universidade Federal Rural de Pernambuco
52.171.900 – Recife – PE – Brasil

[2]Unidade Acadêmica de Garanhuns– Universidade Federal Rural de Pernambuco
55.292-270 – Garanhuns – PE – Brasil

{lucas.sampaio.leite, maasibaldo, tiago.projetos, pmsrodrigo}@gmail.com

Abstract. *This paper presents a game, called Montanha de Chomsky, which aims to give support to the teaching/learning process in the subject of Computing Theory, accompanying the student as a tutor, but in a playful way. Also, allows teachers to create their lists of problems, and as the problems are being solved, are corrected automatically by the application.*

Resumo. *Este artigo apresenta um jogo, denominado Montanha de Chomsky, que tem o objetivo de dar suporte ao ensino/aprendizagem da disciplina de Teoria da Computação. Esta ferramenta acompanha o estudante como um tutor, também permite monitorar o desenvolvimento do aluno. Possibilita aos professores a criação de suas listas de exercício. Os problemas são corrigidos de forma automatica pela aplicação, dando um retorno imediato ao aluno.*

1. Introdução

A Teoria da Computação é fundamental para a Ciência da Computação, não só por proporcionar um adequado embasamento teórico para um correto e amplo entendimento da ciência envolvida na Computação, como também propicia o desenvolvimento lógico e formal necessário em todas as áreas da computação [Aguiar and Oeiras 2010].

Segundo [Gramond and Rodger 1999], existem duas razões pelas quais uma parcela importante dos estudantes têm grandes dificuldade com a disciplina de Teoria da Computação: esta disciplina tem mais matemática que outras da Computação; os estudantes não recebem *feedback* imediato de resolução de problemas.

Uma forma de auxiliar na aprendizagem é fazer uso de jogos educacionais que proporcionem um ambiente de aprendizado atraente e eficaz, devido aos aspectos lúdico e interativo que um jogo pode oferecer [Silva et al. 2010]. Jogos educacionais provocam experiências que o ambiente escolar tradicional não possibilita criar, exigindo do aluno esforços intelectuais e alcançando elevados níveis de aprendizagem comparados aos métodos utilizados nas atividades tradicionais [Pepert 1997]. Além disso, a utilização de Sistemas Tutores pode ser um importante recurso para auxiliar o professor na prática do

*Trabalho apoiado pela Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco (FACEPE) no projeto APQ-0833-1.03/12 *Modelos formais de Computação: alguma teoria e seu ensino*.

ensino-aprendizagem para estudantes de Teoria da Computação. Estes possibilitam uma melhor comunicação entre professores e estudantes, permitindo ao professor acompanhar o desenvolvimento e desempenho do estudante.

Assim, este trabalho apresenta uma ferramenta desenvolvida em *JAVA*,[1] a Montanha de Chomsky que supre a necessidade de, ao propor um ambiente de exercícios para um estudante, obter-se uma correção de forma automática pelo jogo, registrando assim, as informações relevantes do seu aprendizado para fins de acompanhamento. Sua pesquisa é baseada no ensino de Teoria da Computação, estudo de Sistemas Tutores e jogos educacionais. Atua sobre o domínio de Linguagens Regulares e Livres de Contexto.

O presente artigo se divide da seguinte forma. A Seção 2 apresenta trabalhos relacionados. A Seção 3 descreve a construção da Montanha de Chomsky, seguida da arquitetura do sistema tutor, na Seção 4. Na Seção 5 é descrita a metodologia utilizada na apresentação do jogo aos estudantes de Teoria da Computação. As Seções 6 e 7 apresentam os resultados obtidos e as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

A literatura sobre jogos para o ensino de Teoria da Computação é bastante restrita. Podemos citar como referência o Automata Defense 2.0 proposto por [Silva et al. 2010]. Este jogo propõe ao usuário um desafio subjacente: projetar autômatos para vencer o jogo, utilizando de aspectos lúdico e interativo para motivar a aprendizagem do conteúdo. O jogo opera sobre Autômatos Finitos determinísticos (AFD), Autômatos Finitos Não Determinísticos (AFND) e Autômatos com Pilha Derterminísticos (APD).

O método de correção no Automata Defense se dá através da associação de palavras escolhidas aleatoriamente dentro de um conjunto de linguagens formais fixas (6 regulares e 7 livres de contexto), o qual não garante que o autômato tenha sido construído de forma que aceite qualquer palavra de uma determinada linguagem, sendo necessário, posteriormente, a correção de um especialista. Em contrapartida, a Montanha de Chomsky apresenta como diferencial a verificação da corretude dos problemas propostos no âmbito dos modelos que representam linguagens regulares através da equivalência entre dois modelos que representam a mesma linguagem, garantindo, caso o modelo formal do estudante seja equivalente ao modelo resposta para o problema, o reconhecimento de qualquer palavra de uma determinada linguagem regular.

No âmbito dos simuladores, podemos destacar o já bastante conhecido Java Formal Languages and Automata Package, [Rodger and Finley 2006], desenvolvido em JAVA, para experimentação de diversos tópicos de autômatos e gramáticas, incluindo implementações de Expressões Regulares (ER), AFDs, AFNDs, Autômatos com Pilha (AP), Máquinas de Mealy e Moore, Máquinas de Turing (MT), além de fazer diversas conversões entre os modelos que possuem equivalência. Vale ressaltar que o método de correção dos modelos formais construídos pelo usuário consiste no processamento de palavras de entrada, fornecidas pelo usuário, o que não permite verificar se a linguagem esperada é aquela representada.

O LabLF, proposto por [Aguiar and Oeiras 2010], apresenta uma forma diferente

[1]Ainda restrita ao uso dos autores, mas a ser disponibilizada publicamente após etapas de desenvolvimento e testes suplementares.

de verificar a corretude de autômatos em relação às ferramentas anteriormente citadas e bastante parecida com o método de correção deste trabalho. O LabLF reutiliza os editores da ferramenta computacional JFLAP para implementação de AFD, AFND, AP e MT, porém, a verificação é dada de forma automática. O LabLF permite que o usuário teste manualmente os autômatos construídos através de palavras de entrada, porém, após submetida a questão, a aplicação devolve automaticamente se o resultado da questão é o esperado pelo professor. O método de correção é feito através da conversão para o AFD mínimo a fim de ser comparado com o autômato-resposta da questão.

Para os exercícios em que as soluções são modelos de AP e/ou MT, cujos respectivos problemas de equivalência são indecidíveis, o processo de correção é feito através de palavras testes, o que não garante a veracidade da questão. Fazendo uma breve comparação entre a ferramenta LabLF e a Montanha de Chomsky, podemos ver o sistema tutor como principal diferencial desta segunda, possibilitando suporte ao professor quanto as informações de aprendizado de cada aluno. Outros pontos diferenciais são a possibilidade de criação de problemas tanto pelos estudantes, quanto pelos professores para resoluções posteriores e a resolução dos problemas da Montanha de Chomsky serem efetuadas de forma lúdica.

Proposto por [Dognini and Raabe 2003] como ferramenta para auxiliar a aprendizagem de linguagens regulares, o EduLing é dividido em três módulos: Tutorial, Experimentação Livre e Desafio. Embora o módulo tutorial aborde os conceitos fundamentais sobre a teoria das Linguagens Regulares, este não captura informações relevantes sobre o desempenho de cada estudante, como acontece na Montanha de Chomsky, a fim de monitoramento e auxílio às atividades do professor. O aplicativo aqui apresentado também se diferencia pela abordagem das Linguagens Livres de Contexto.

A Tabela 1 organiza as ferramentas acima citadas, mostrando a composição de cada uma comparada com a Montanha de Chomsky. A correção automática tratada em questão não é uma verificação apenas por aceitação de palavras de entrada, mas sim através da verificação de equivalência entre modelos em que esse caso é possível.

Tabela 1. Ferramentas e suas funcionalidades: LR (Linguagens Regulares), LLC (Linguagens Livres de Contexto), IG (Interface Gráfica), Correção automática (feedback imediato), Sistema Tutor, Aprendizagem Lúdica

	LR	LLC	IG	C. automática	S. Tutor	A. Lúdica
Automata Defense 2.0	X	X	X			X
JFLAP	X	X	X			
LabLF	X	X	X	X		
EduLing	X		X		X	
Montanha de Chomsky	X	X	X	X	X	X

3. A Montanha de Chomsky

Em linhas gerais, o jogo consiste numa sequência de ambientes – ou fases – seguindo a Hierarquia de Chomsky, onde os três primeiros ambientes são pertencentes às linguagens do Tipo 3 (Linguagens Regulares) e um quarto Ambiente às linguagens do Tipo 2 (Linguagens Livres de Contexto). Em cada ambiente, o jogador se depara com problemas referentes às linguagens da família em questão. Tais problemas são encarados como

inimigos que o usuário precisa vencer, tarefa que consiste na construção de um modelo formal – ou autômato, ou expressão regular, ou gramática, a depender do ambiente em que o usuário se encontra – que represente a linguagem em questão (ou que reconheça trechos de uma linguagem de programação, no caso do ambiente 4). Cada ambiente do jogo possui 20 estágios.

O objetivo é alcançar o topo da montanha através da resolução de problemas. Ao passo que o usuário derrota os inimigos – resolve os problemas, novos estágios são desbloqueados, e os respectivos novos problemas são mais desafiadores, pois o nível de conhecimento e interpretação exigido aumenta. Ao resolver todos os problemas propostos de uma família (ambiente), um novo ambiente mais amplo é liberado, e o usuário, à medida que os problemas se tornam mais difíceis, aproxima-se do topo da montanha, percorrendo a hierarquia de Chomsky, justificando assim o nome do jogo.

3.1. Ambiente 1

Inicialmente, apenas o Ambiente 1 é liberado ao usuário. Este é composto por problemas que envolvem a representação de uma Linguagem Regular através de uma expressão regular (ER). É dado ao usuário uma descrição informal quanto à linguagem regular a ser representada e apresentadas as formas mais básicas da estrutura de uma expressão regular reconhecível pelo jogo, mostrando como aplicar as operações elementares de construção de uma ER, como a união, concatenação e estrela de Kleene. Além de exemplos de construção de ERs, também é disponível na tela de resolução material de apoio com todo conteúdo referente à teoria necessária para realização do problema proposto.

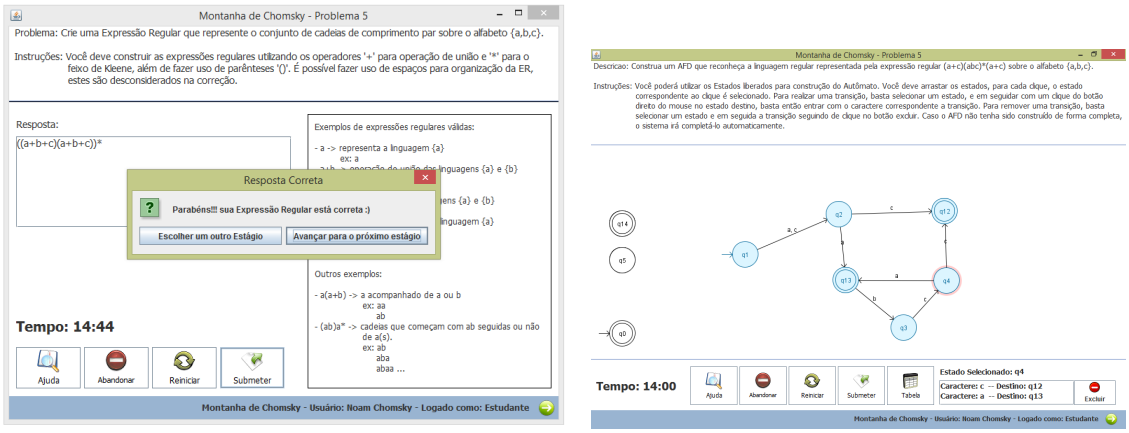
Ao final da construção da expressão regular, o usuário deve submeter a sua resposta para correção conforme mostra a Figura 1(a). A verificação da corretude do problema é baseada na verificação de equivalência entre dois AFDs, que consiste simplesmente em uma verificação dos estados finais da parte acessível do produto cartesiano dos autômatos em questão (portanto em complexidade polinomial). Há diversos algoritmos clássicos para a conversão da expressão regular em um AFD equivalente, que podem ser consultados nas referências principais da área. Em nosso jogo, o primeiro passo é a conversão entre uma expressão regular em um AFND equivalente usando o algoritmo de Thompson [Thompson 1968] que resulta em um AFND com ε-transições com apenas um estado inicial e um final. Realizada essa transformação basta eliminar as ε-transições e construir o AFD a partir da construção dos subconjuntos.

3.2. Ambiente 2

Em cada estágio do ambiente 2, o usuário está diante de uma descrição de uma linguagem reconhecível por uma expressão regular. Sua tarefa consiste na construção de um AFD equivalente à linguagem representada. Essa construção requer a "montagem" do autômato utilizando peças que são fornecidas pelo jogo, no caso os estados e transições. Na Figura 1(b) é apresentada a representação gráfica de um estágio no qual é dada uma descrição do problema, instruções e estados disponíveis para a construção do autômato.

A ferramenta dispõe durante o tempo de realização do desafio proposto a visualização da tabela de transições do autômato sendo construído pelo usuário.

Ao término da elaboração da resposta deste estágio, o usuário deve submeter sua resposta ao jogo para verificação da equivalência com o autômato resposta em questão.



(a) (b)

Figura 1. (a) Construção de Expressões Regulares (Ambiente 1); (b) Construção de um AFD (Ambiente 02)

Feito isso é dado um feedback imediato ao usuário com (i) a aceitação e avanço ao próximo estágio – o que indica que o usuário construiu um modelo formal equivalente ao modelo resposta que expressa a linguagem regular pedida – tornando mais próximo do objetivo do jogo que é chegar ao topo da montanha; ou com (ii) a rejeição da sua resposta, caso o modelo construído não reconheça a linguagem regular representada pela expressão regular descrita no problema do estágio.

3.3. Ambiente 3

No âmbito dos Autômatos Finitos não determinísticos e AFNDs com ϵ- transições, o Ambiente 3 têm representação bastante similar ao Ambiente 2 e as regras e disponibilizações de material de apoio e visualização de modelo formal se mantêm. No entanto, a tarefa do usuário consiste agora na construção de um AFND equivalente à linguagem representada.

Para os autômatos não determinísticos, a solução consiste primeiramente na eliminação das ϵ-transições – caso possua, gerando um AFND equivalente [Menezes 2011]. Em seguida, o AFND já sem as ϵ-transições é determinizado através da construção dos subconjuntos [Hopcroft et al. 2002], que inclui a construção de todos os subconjuntos do conjunto de estados do AFND, obtendo o AFD equivalente, o qual é corrigido utilizando o mesmo algoritmo verificador de equivalência entre dois AFDs utilizado para o Ambiente 2.

Uma das dificuldades encontradas durante esta tarefa é a explosão de tamanho em ordem exponencial onde apesar de ser bastante comum na transformação de um AFND em AFD que o AFD tenha aproximadamente o mesmo número de estados do AFND a partir de qual é construído, o crescimento exponencial no número de estados é possível, consistindo no pior caso em todos os 2^n estados do AFD construídos através do AFND de n estados se mostrarem acessíveis. Para contornar tal situação, foram escolhidos problemas para resolução que não necessitam de um número grande de estados na construção do modelo formal, evitando assim um esforço computacional exagerado na fase de verificação da resposta.

3.4. Ambiente 4

No âmbito das Linguagens Livres de Contexto, ou tipo 2, a forma de construção dos problemas não pode ser tratada como nos ambientes que representavam linguagens regulares. O principal motivo se dá por não existir um algoritmo que verifique a igualdade de duas linguagens livres do contexto [Menezes 2011]. Um outro empecilho é o fato de que não se pode determinar computacionalmente se uma gramática livre de contexto é ambígua [Menezes 2011] [Hopcroft et al. 2002].

Diante dos problemas expostos, o Ambiente 4 proporciona ao usuário a construção das regras de produção de uma Gramática Livre de Contexto (GLC) que reconheça pequenos trechos de códigos de linguagens de programação, o que faz com que a resposta do problema seja uma GLC restrita àquelas entradas. Pede-se ao usuário que construa as regras de produção sem o uso de ambiguidades. No entanto, antes da correção do problema, a gramática é fatorada à esquerda como forma de eliminar a indecisão sobre qual produção aplicar quando duas ou mais produções iniciarem com a mesma forma sentencial. Com a gramática fatorada à esquerda, é feita então o reconhecimento através da análise descendente preditiva tabular com auxílio dos conjuntos *FIRST* e *FOLLOW*.

Um *feedback* é dado ao usuário após o reconhecimento ou não dos trechos de linguagem de programação referentes ao problema. Além do *feedback*, é dado também ao usuário os movimentos executados pelo analisador preditivo tabular, contendo todas as alterações efetuadas na pilha, na entrada e quais produções foram expandidas durante as alterações como mostra a Figura 2.

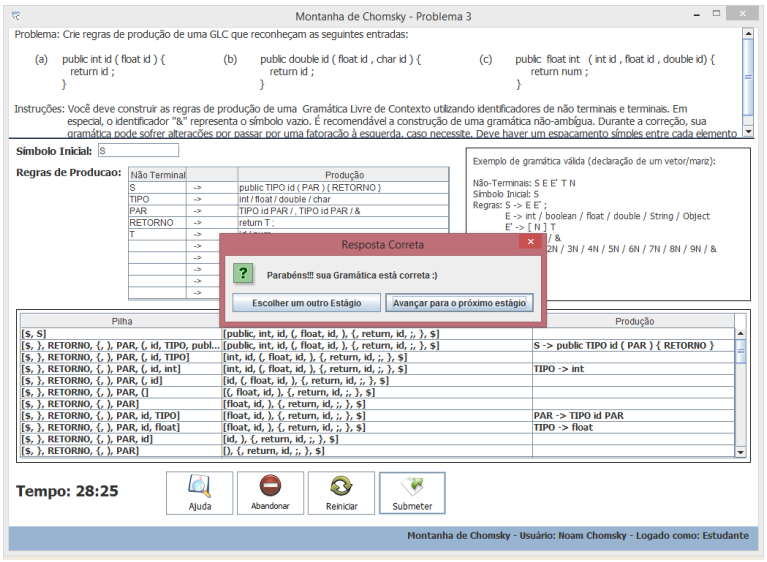


Figura 2. Reconhecimento de uma Linguagem Livre de Contexto (Ambiente 4)

3.5. Criação de problemas

Uma das funcionalidades do jogo é, de forma alternativa à resolução apenas dos problemas nativos da ferramenta, permitir aos estudantes e principalmente ao professor, a criação de problemas para serem respondidos posteriormente. Esta funcionalidade possibilita ao professor passar suas listas de exercício diretamente no jogo, fazendo com

que seus alunos tenham o material necessário para respondê-las, além do *feedback* dos problemas.

A criação de problemas que lidam com a classe de linguagens regulares é realizada apenas através de uma descrição do problema, alfabeto a ser utilizado e a inserção de uma expressão regular que é convertida pela aplicação para um AFD resposta. Para criação de problemas que envolvem as Linguagens Livres de Contexto, basta fornecer como entrada trechos de linguagens livres de Contexto a serem geradas pelas regras de produção construídas pelo usuário.

4. O Sistema Tutor

O Sistema tutor da Montanha de Chomsky implementa os modelos do tutor, do domínio e do estudante conforme a arquitetura básica de um Sistema Tutor proposta por [Raabe 2005]. O modelo do tutor é responsável por requisitar ao modelo do domínio o material que se deseja ensinar ao estudante e o problema a ser resolvido em determinado estágio. Este modelo ainda informa as instruções e orientações de como responder cada estágio e apresenta as ferramentas para construção de autômatos, expressões regulares e regras de produções de Gramáticas Livres de Contexto, além de avaliar as soluções submetidas pelo estudante. O controle também é responsável pelo preenchimento do modelo do estudante, guardando as informações relevantes sobre o seu aprendizado.

O modelo do domínio é representado como a apostila com todo o conteúdo teórico preciso para responder os problemas. Esta é disponibilizada para suporte teórico ao estudante. A estratégia de ensino também é um componente desse modelo, assim como a divisão de problemas por ambientes de acordo com os assuntos abordados dentro da Teoria da Computação.

Por último, têm-se o modelo do estudante onde são encontradas informações relativas ao aprendizado de cada estudante, tais como percentuais de acertos e erros, detalhamento das questões respondidas com êxito ou não, entre outras informações relevantes adquiridas durante a utilização do jogo.

Através da interface o usuário estabelece uma comunicação com o sistema tutor. A cada estágio são expostos os problemas, instruções e material de apoio que varia do estado em que o estudante se encontra no jogo. A interface de usuário também proporciona a visualização de desempenho do estudante, onde dependendo do perfil em que esteja sendo acessado são obtidas as informações. Para o perfil do estudante, é possível apenas a visualização do desempenho do mesmo, enquanto o professor poderá acessar informações referentes a todos os estudantes que utilizaram o jogo até aquele momento como mostra a Figura 3.

O professor também pode obter relatórios detalhados, sendo possível realizar filtragem sobre os ambientes e requisitos que deseje obter dados tais como as questões respondidas corretamente (data, hora, tempo de resposta, número do estágio, ambiente), abandonos de estágios, visualização do material de apoio, sendo detalhada a data, hora, entre outras ações realizadas pelos estudantes.

5. Apresentando a Montanha de Chomsky aos estudantes

Os métodos utilizados na coleta de dados em estudo de usuários estão relacionados com tipo de abordagem qualitativa ou quantitativa, sendo que os questionários são

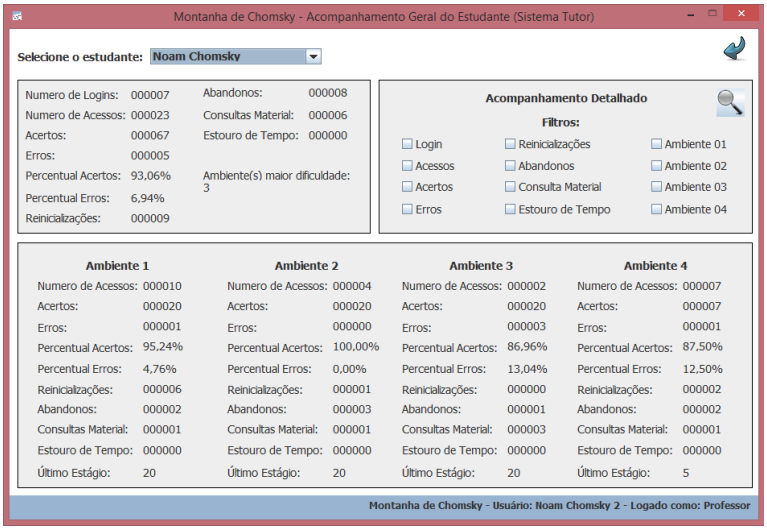


Figura 3. Interface gráfica do sistema tutor

utilizados em estudos quantitativos, mas que podem ter questões abertas que coletem dados qualitativos [Baptista and Cunha 2007].

As principais vantagens da utilização de questionários na coleta de dados são apontadas por [da Cunha 1982] como sendo um método que obtém respostas rápidas e precisas, de baixo custo e que permite atingir uma grande população dispersa, dando maior grau de liberdade e tempo ao respondente. Segundo [Perkins 2004], com a expansão da Word Wide Web (WWW), o questionário passou a adquirir uma importância maior em relação aos outros instrumentos por estar disponível em qualquer computador conectado a internet independente de tempo e local, possibilitando a inserção de textos, imagens e sons a um menor tempo de envio e recebimento das respostas, bem como sua transcrição a uma base de dados que possibilite análise estatística dos resultados.

Foi realizada uma experiência entre estudantes com o jogo, sendo imposto o pré-requisito de o avaliado ter cursado ou estar em curso na cadeira de Teoria da Computação ou correlata que tenha o conteúdo exigido para uso do jogo. Como método de coleta de dados, foi utilizada a aplicação de questionário eletrônico com o objetivo de observar a viabilidade da adoção do jogo durante cursos de Teoria da Computação como ferramenta de suporte ao ensino-aprendizagem dos estudantes.

O cenário de uso da Montanha de Chomsky foi realizado nos cursos de Bacharelado em Ciência da Computação e Licenciatura em Computação da UFRPE. Primeiramente foram expostos os tópicos de abordagem do jogo e em seguida foram liberadas as máquinas para que o estudante tivesse contato com o *software*. Participaram da experiência trinta e nove (39) estudantes voluntários, sendo dezenove (19) estudantes matriculados na cadeira de Teoria da Computação e dez (10) que já haviam cursado esta cadeira.

Ao final da experiência com o jogo, todos os estudantes responderam um questionário composto por doze (12) questões, distribuídas na forma de múltipla escolha, escalas de classificação e dissertativas, abordando três pontos principais a serem avaliados: i) a experiência do uso do jogo; ii) o auxílio no processo de ensino-

aprendizagem, e ainda iii) a concordância dos problemas com o conteúdo apresentado na sala de aula. Por último foi aberto espaço para críticas e sugestões para melhorias e trabalhos futuros a fim de serem levantadas algumas questões para melhorias na ferramenta e trabalhos e estudos futuros.

6. Análise dos resultados

Diante a experiência vivida pelos estudantes com o jogo, foram obtidos resultados através do formulário de coleta de dados. Primeiramente foi pedido aos estudantes que atribuíssem uma nota em uma escala de 0 (zero) a 10 (dez) quanto ao uso da ferramenta, onde 41% atribuíram a nota máxima: 10 (dez), seguido de 26% nota 9 (nove), 18% nota 8 (oito) e 15% atribuíram 7 (sete) como nota.

Quanto às dificuldades apresentadas pelos estudantes durante a utilização da ferramenta, 38% tiveram algum tipo de dificuldade, justificando ser ocasionada pelo fato de ser o primeiro acesso dos mesmos com a ferramenta e pelas tentativas de resposta dos estágios sem a leitura prévia das instruções. Foram relatadas também dificuldades quanto ao entendimento em algumas questões, onde parte dos alunos declararam não ter conhecimento da notação formal utilizada nos enunciados dos estágios.

Perguntado se os problemas expostos nos estágios eram condizentes com os vistos nos exercícios em sala de aula, apenas um (1) aluno respondeu negativamente, o que equivale a apenas 3% do total de estudantes. Vale ressaltar que as questões utilizadas durante a execução do experimento foram retiradas de exercícios de livros clássicos na área de Teoria da Computação, tais como [Menezes 2011] [Sipser 2007] [Hopcroft et al. 2002]. Sendo o principal objetivo deste trabalho o auxílio no processo do ensino/aprendizagem da disciplina de Teoria da Computação, perguntado aos estudantes se a utilização do jogo simultaneamente ao estudo da disciplina facilitaria o processo de ensino-aprendizagem da disciplina, 100% responderam positivamente e alegaram posteriormente, que o jogo teria ajudado bastante principalmente nos seus estudos para realização das avaliações, pelo fato de possuir um *feedback* imediato quanto as possíveis soluções dos problemas.

Questionado se os estudantes tinham dificuldades em responder problemas relacionados a disciplinas de teoria da computação de maneira tradicional, foi obtido um percentual de 74% responderam positivamente, onde 55% deste percentual afiram ter dificuldades devido a correção no método tradicional através de testar o reconhecimento através de soletrar palavras, enquanto 45% apontaram ter dificuldades ocasionadas por falta de *feedback* imediato ao responder os problemas. Diante desta dificuldade, trinta e quatro alunos (34), equivalente a 87% afirmaram preferir efetuar a resolução de problemas relacionados a Teoria da Computação através do jogo.

7. Conclusões e trabalhos futuros

A Montanha de Chomsky foi criada para ser utilizada tanto por professores quanto por alunos, com o objetivo de facilitar o processo de ensino/aprendizagem da disciplina de Teoria da Computação. Do ponto de vista do aluno, por aprender jogando de forma dinâmica e interativa, e do professor por dispor de um sistema com a arquitetura de sistema tutor que trata informações importantes quanto ao desempenho e aprendizado dos estudantes, além de dispor também da possibilidade de criação de listas de exercícios.

Após a experiência dos alunos de Teoria da Computação com o jogo em questão, foi verificado que o mesmo pode vir a colaborar fortemente no processo de ensino/aprendizagem visto que 100% dos estudantes a avaliaram positivamente. Outra vantagem é sua independência de plataforma, por ser desenvolvida em JAVA.

Como trabalhos futuros, propõe-se a adição de mais ambientes dentro das Linguagens Livres de Contexto podendo trabalhar com Autômatos de Pilha não determinísticos e a inserção de questões para Linguagens Sensíveis ao Contexto e as Recursivamente Enumeráveis, trabalhando com seus formalismos geradores (Gramáticas Sensíveis ao Contexto) e reconhecedores (Máquinas de Turing).

Referências

Aguiar, R. and Oeiras, J. Y. (2010). Laboratório de linguagens formais. *Revista Brasileira de Informática na Educação*, 18(1):106.

Baptista, S. G. and Cunha, M. B. (2007). Estudo de usuários: visão global dos métodos de coleta de dados. *Perspectivas em Ciência da Informação*, 12(2):168–184.

da Cunha, M. B. (1982). Metodologias para estudo dos usuários de informação científica e tecnológica. *Revista de Biblioteconomia de Brasília*, 10(2):5–19.

Dognini, M. J. and Raabe, A. L. A. (2003). Eduling - software educacional para linguagens regulares. *Anais do XIV Brasileiro de Informática na Educação*, pages 216–225.

Gramond, E. and Rodger, S. H. (1999). Using jflap to interact with theorems in automata theory. *SIGCSE Bull.*, 31(1):336–340.

Hopcroft, J. E., Ullman, J. D., and Motwani, R. (2002). *Introdução à teoria dos autômatos, linguagens e computação*. Elsevier. 165–166.

Menezes, P. B. (2011). *Linguagens formais e autômatos*. Porto Alegre: Bookman.

Pepert, S. (1997). *Looking at Technology Through School-Colored Spectacles*. Logo Exchange.

Perkins, G. H. (2004). Will libraries' web-based survey methods replace existing non-electronic survey methods? *Information technology and libraries*, 23(3):123–126.

Raabe, A. L. A. (2005). Uma proposta de arquitetura de sistema tutor inteligente baseado na teoria das experiências de aprendizagem mediadas. 152p. Tese (Doutorado)-Curso de Pós-graduação em Informática Na Educação. Universidade Federal do Rio Grande do Sul. Porto Alegre.

Rodger, S. H. and Finley, T. W. (2006). *An Interactive Formal Languages and Automata Package*. Jones Bartlett Publishers, Sudbury, MA.

Silva, R. C., Binsfeld, R. L., Carelli, I. M., and Watanabe, R. (2010). Automata defense 2.0: reedição de um jogo educacional para apoio em linguagens formais e autômatos. *Anais do XXI Simpósio Brasileiro de Informática na Educação*.

Sipser, M. (2007). *Introdução à Teoria da Computação*. 2ª ed. São Paulo: Cengage Learning.

Thompson, K. (1968). Programming techniques: Regular expression search algorithm. *Communications of the ACM*, 11(6):419–426.