Camilo Peric de Freitas

Elaboração de um jogo de expressões algébricas

São Paulo, SP

2014

Camilo Peric de Freitas

Elaboração de um jogo de expressões algébricas

Escola Superior de Engenharia e Gestão - ESEG
Curso de Graduação em Sistemas de Informação

Orientador: Marcelo Novaes de Rezende

São Paulo, SP
2014

Camilo Peric de Freitas
Elaboração de um jogo de expressões algébricas/ Camilo Peric de Freitas. – São Paulo, SP, 2014-
95 p. : il. (algumas color.) ; 30 cm.

Orientador: Marcelo Novaes de Rezende

Trabalho de Graduação – Escola Superior de Engenharia e Gestão - ESEG
Curso de Graduação em Sistemas de Informação , 2014.
1. Palavra-chave1. 2. Palavra-chave2. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

CDU 02:141:005.7

Camilo Peric de Freitas

Elaboração de um jogo de expressões algébricas

Trabalho aprovado. São Paulo, SP, 24 de novembro de 2012:

Marcelo Novaes de Rezende
Orientador

Professor
Convidado 1

Professor
Convidado 2

São Paulo, SP
2014

Este tabalho eu dedico a todos aqueles que fizeram parte do excelente Curso de Graduação em Sistemas de Informação da Escola Superior de Engenharia e Gestão. A minha graduação acaba assim como o curso, mas tudo o que foi construído durante esta jornada eu levarei comigo.

Aos professores a gratidão é imensa, o que vocês me passaram excede em muito todo o investimento envolvido na conclusão do curso de graduação.

Agradecimentos

Os agradecimentos principais são direcionados à Gerald Weber, Miguel Frasson, Leslie H. Watter, Bruno Parente Lima, Flávio de Vasconcellos Corrêa, Otavio Real Salvador, Renato Machnievscz[1] e todos aqueles que contribuíram para que a produção de trabalhos acadêmicos conforme as normas ABNT com LaTeX fosse possível.

Agradecimentos especiais são direcionados ao Centro de Pesquisa em Arquitetura da Informação[2] da Universidade de Brasília (CPAI), ao grupo de usuários *latex-br*[3] e aos novos voluntários do grupo *abnTeX2*[4] que contribuíram e que ainda contribuirão para a evolução do abnTeX2.

[1] Os nomes dos integrantes do primeiro projeto abnTeX foram extraídos de <http://codigolivre.org.br/projects/abntex/>

[2] <http://www.cpai.unb.br/>

[3] <http://groups.google.com/group/latex-br>

[4] <http://groups.google.com/group/abntex2> e <http://abntex2.googlecode.com/>

"Não vos amoldeis às estruturas deste mundo,
mas transformai-vos pela renovação da mente,
a fim de distinguir qual é a vontade de Deus:
o que é bom, o que Lhe é agradável, o que é perfeito.
(Bíblia Sagrada, Romanos 12, 2)

Resumo

Este trabalho tem como tema a utilização de Tecnologias da Informação como ferramentas de ensino. O objetivo é o desenvolvimento de um experimento que tem como produto final um protótipo de jogo de expressões algébricas. O jogo gera problemas, permite a resolução dos mesmos além de avaliá-los para encontrar todas as soluções possíveis.

Palavras-chaves: Tecnologias da Informação. educação.

Abstract

This work's theme is the use o of Information Technologies as a tool for teaching. The objective is the development of an experiment which has as outcome a algebraic expressions game prototype. The game generate problems, allows the resolution of these besides evaluate them to find all the possible solutions.

Key-words: Information Tecnologies. education.

Lista de ilustrações

Sumário

Introdução

0.1 Objetivo

O objetivo deste trabalho é fazer um experimento cujo tema é Tecnologias da Informação e Educação. O experimento consiste no desenvolvimento de um sistema que possa servir como uma ferramenta de ensino de um determinado conteúdo escolar. O sistema proposto é um jogo de expressões algébricas desenvolvido utilizando a tecnologia HTML5, a quinta versão do padrão HTML. O jogo consiste em apresentar expressões algébricas para o jogador resolver. Assim que o sistema apresentar uma expressão ao usuário este deve selecionar uma das operações contidas na expressão para resolvê-la e então o fazer. Caso a expressão resultante ainda possuir operações para serem resolvidas os passos anteriores se repetem, até que não existam mais operações na expressão resultante significando que o usuário chegou ao resultado final daquela expressão. O jogo deve possuir um conjunto de expressões em sua base de dados, que devem envolver as operações de soma, subtração, multiplicação e divisão, além de ser capaz de gerar expressões aleatórias. As expressões geradas não devem involver multiplicações e divisões para que o escopo do experimento não seja extenso.

0.2 Motivação

A motivação deste trabalho é explorar a possibilidade de melhorar a educação através da utilização de Tecnologias da Informação. Com sistema capazes de gerar problemas, permitir o desenvolvimento da resolução de tais problemas e avaliar a resolução feita pelo aluno existe a possibilidade de guardar informações que podem ser relevantes para uma análise do professor, tanto sobre as dificuldades dos alunos quanto sobre possíveis pontos falhos em sua metodologia e/ou didática. As informações relevantes que podem ser armazenadas devem estar estruturadas para que a possibiliade de análise dos dados seja maior. Tais informações podem incluir erros mais comuns, quantidade de exercícios feitos, aproveitamento para um determninado tópico, entre muitos outros. Tais informações geradas ainda podem retroalimentar o sistema, por exemplo, o jogo pode escolher o nível de dificuldade do problema para determinado aluno ou escolher determinados tipos de problemas que o aluno tenha menor aproveitamento, entre outras possibilidades. O ensino a distância já demonstra um esforço no sentido de utilizar Tecnologias da Informação na educação. De acordo com o AbraEAD (Anuário Brasileiro Estatístico de Educação Aberta a Distância) o número de estudantes que fizeram cursos com metodologia a distância em 2007 foi dois milhões e meio. O número de instituições credenciadas pelo Sistema de Edu-

cação em 2008 foi 972.826. As instituições credenciadas incluem desde ensino fundamental até pós-graduação. (ABRAED) A plataforma online de ensino Code School1 que ensina diversas habilidades em programação e web design já implementa cursos2 que contém funcionalidades semelhantes às do experimento proposto. Os exercícios não são criados aleatóriamente pela plataforma mas a mesma permite o desenvolvimento da solução e é capaz de avaliar a solução dada pelo usuário, o que possibilita que alunos façam o curso e sejam avaliados sem a necessidade de alocar uma pessoa para fazer a avaliação. O site Edudemic3 cujo slogan em português é "conectando educação e tecnologia" e tem como meta conectar as melhores tecnologias no planeta a professores, administradores e alunos, entre outros possui uma lista (EDUDEMIC) de 50 ferramentas educacionais tecnológicas. Entre elas vale a pena destacar o site fundado por Salman Khan, Khan Academy4. O site conta com diversos cursos em vídeo-aula que vão desde exatas, como matemática e ciências, até humanidades e artes, como história e música respectivamente. A escolha da tecnologia HTML5 foi motivada pela gama de dispositivos que possuem suporte para a mesma, que vai de PCs a Smart Tvs, incluindo Smartphones. O amplo suporte à tecnologia utilizada permite que o jogo desenvolvido seja acessível em um número maior de dispositivos além de, por exemplo, permitir que os usuários possam fazer exercícios em qualquer lugar caso posssuam um Smartphone, sem depender de um PC.

0.3 Organização

No primeiro capítulo a tecnologia HTML5, a principal utilizada neste trabalho, é abordada. No capítulo seguinte as outras tecnologias utilizadas são introduzidas. Por último o texto que explica o experimento desenvolvido neste trabalho. No capítulo 1 a tecnologia HTML5 é apresentada. Além de uma explicação da versão 5 do padrão HTML (HTML5) e suas características o padrão HTML em sí é abordado históricamente mostrando sua evlução desde seu surgimento. Em Tecnologias Utilizadas as tecnologias complementares ao HTML5, utilizadas no experimento, são explicadas em âmbitos gerais. Tais tecnologias envolvem CSS3 e JavaScript para a apresentação e dinamicidade respeticamente, o framework PhoneGap utilizado para construir a aplicação móvel, a ferramenta de controle de versão Git entre outras. Na parte final do texto o experimento é explicado partindo da visão geral. O principal problema envolvido é abordado em detalhes nos tópicos Criação de expressões e Análise de soluções.

0.4 Metodologia

O primeiro passo na elaboração deste trabalho foi escolher o conteúdo educacional que seria abordado levando em consideração o tamanho do escopo que o trabalho deve ter. O conteúdo escolhido a ser trabalhado é expressões algébricas. A plataforma escolhida para

o desenvolvimento inclui HTML5, CSS3, JavaScript, Queue.js, Highcharts e PhoneGap. A análise do problema envolvido no experimento foi feita com fim de compreender a natureza do mesmo. A criação de expressões e a avaliação de soluções para expressões são os pontos críticos envolvidos no problema. Uma pesquisa sobre criação de expressões algébricas esclarece que tais expressões algébricas podem ser obtidas através da utilização de uma gramática formal. Além de gerar expressões algébricas o sistema deve ser capaz de conhecer suas possíveis soluções. A forma escolhida para encontrar soluções partiu de estudos sobre pequenas expressões algébricas e suas possíveis soluções, os quais expuseram semelhanças entre as expressões, representadas como árvores, e árvores binárias ordenadas. O desenvolvimento do jogo foi orientado a comportamento, de tal forma que a diretriz estabelecida para o desenvolvimento foi o comportamento esperado da aplicação. O comportamento da aplicação se espelha na forma como os alunos resolvem expressões algébricas. Os testes da aplicação foram feitos tanto de maneira manual quanto automatizada. Os testes automatizados foram utilizados apenas sobre a função de avaliação de soluções no sentido de conhecer o tempo de avaliação das árvores. Os demais testes, funcionais, foram todos feitos manualmente.

Parte I

Referenciais teóricos

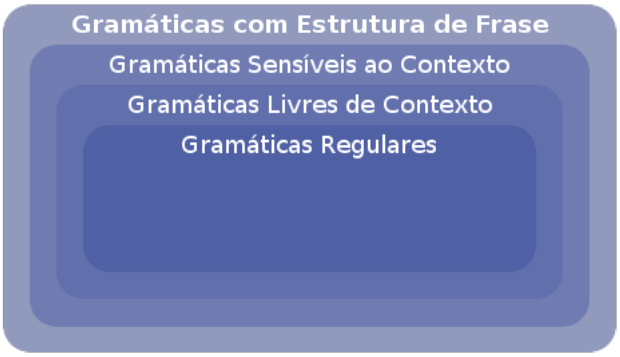
1 Linguagens e gramáticas

Uma linguagem é um conjunto abitrário de cadeias - palavras - sobre um alfabeto (GLADKII,). Uma das opções para a representação finita de uma linguagem é a utilização de uma gramática. A noção formal de gramática foi introduzida por Noam Chomsky na década de 50. Uma gramática é definida como uma tupla que contém: um conunto finito de símbolos terminais, um conjunto finito de símbolos não-terminais, um estado inicial (pertencente ao conjunto dos símbolos não-terminais) e um conjunto finito de produções. Os conjuntos de símbolos terminais e não-terminais são disjuntos, ou seja, não possuem elementos em comum. (ARTALE, 2014)

1.1 Classificação de Chomsky

Toda gramática pode ser classificada de acordo com a Classificação de Chomsky onde toda gramática é pelo menos uma Gramática com Estrutura de Frase. A classe seguinte é um subconjunto da anterior e contém as Gramáticas Sensíveis ao Contexto. A classe das Gramáticas Livres de Contexto é novamente um subconjunto da classe anterior, assim como a ultima classe das Gramáticas Regulares é um subconjunto desta. A figura a seguir expressa a hierarquia existente entre as classes. A Classificação de Chomsky leva em conta como são as produções da gramática.

Figura 1 – Classes da Classificação de Chomsky e sua hierarquia



Fonte: Produzido pelo autor

Lembrando que as gramáticas são compostas por: um conjunto finito de símbolos não-terminais N, um conjunto finito de símbolos terminais T, um conjunto de produções P e um símbolo inicial σ. A intersecção entre os conjuntos de símbolos não terminais e

terminais deve ser vazia. O conjuto de produções é um subconjunto de todas as produções possíveis, o produto cartesiano de: todas as palavras geradas com símbolos não-terminais e terminais contendo pelo menos um símbolo não terminal (lado esquerdo da produção); todas as palavras geradas com símbolos não-terminais e terminais. O símbolo inicial deve pertencer ao conunto dos símbolos não terminais. Nas definições a seguir λ denota a palavra nula. (LERMA,)

1.1.1 Gramáticas com Estrutura de Frase

A gramática é com Estrutura de Frase se todas as produções são da forma:

$\alpha \to \delta$, onde $\alpha \in (N \cup T)^*$ - T^*, $\delta \in (N \cup T)^*$.

1.1.2 Gramáticas Sensíveis ao Contexto

A gramática é Sensível ao Contexto se todas as produções são da forma:

$\alpha A \beta \to \alpha \delta \beta$, onde $\alpha, \beta \in (N \cup T)^*, A \in N, \delta \in (N \cup T)^*$ - $\{\lambda\}$.

1.1.3 Gramáticas Livres de Contexto

A gramática é Livre de Contexto se todas as produções são da forma:

$A \to \delta$, onde $A \in N, \delta \in (N \cup T)^*$.

1.1.4 Gramáticas Regulares

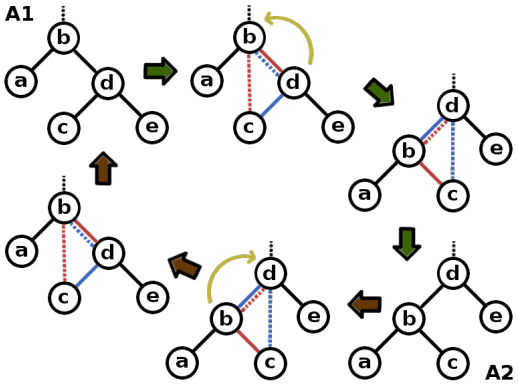
A gramática é Regular se todas as produções são da forma:

$A \to a$ ou $A \to aB$ ou $A \to \lambda$.

2 Árvores binárias ordenadas e rotações

Uma árvore binária é uma coleção de dois tipos de nós, externos e internos, e três tipos de relações entre estes nós: pai, filho à esquerda e filho à direita. Todos os nós têm um nó pai, exceto um que é chamado de raíz. (SLEATOR; TARJAN; THURSTON, 1988)

Figura 2 – Diagrama sobre rotação de árvores binárias ordenadas



Fonte: Produzido pelo autor

O diagrama contido na Figura 3 mostra graficamente como as rotações anti-horária e horária acontecem. A rotação anti-horária é explicada no fluxo representado através das setas verdes, que começa na árvore A1 e termina na árvore A2. A rotação horária de uma forma simétrica começa na árvore A2 e termina na árvore A1, as setas marrons representam o fluxo. As setas amarelas mostram o sentido das rotações executadas.

A rotação horária parte da árvore A2. A árvore seguinte mostra que o nó b, que inicialmente tem como filho direito o nó c, terá em seguida o nó d como seu filho direito. O filho esquerdo do nó d será alterado de nó b para nó c. A possível relação entre o nó d e o seu nó pai será passada para o nó b. Em seguida observamos como linhas contínuas as novas ligações e como tracejadas as antigas. Por fim chegamos a árvore A1 que é o resultado final da rotação horária sobre o nó d da árvore A2.

A rotação anti-horária é um processo simétrico à rotação horária. A sequência expressa através das setas verdes, de A1 até A2, explica a transformação da mesma forma como a sua transformação inversa explicada no tópico anterior sendo assim não será explicada em detalhes.

3 Propriedades das operações

4 Números de Catalan

Os números de Catalan é um conjunto de números que aparece em probelmas de enumeração de árvores como no problema de divisão dos polígonos de Euler. O n-ésimo número de Catalan pode ser obtido através da expressão $\frac{1}{n+1}\binom{2n}{n}$ (STANLEY; WEISSTEIN,). Os dez primeiros números da sequência, começando com $n = 0$, são: 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862.

Os números de Catalan aparecem em diversos problemas, entre eles o problema das Ordens de Mutiplicação. Este problema se refere a uma sucessão de n multiplicações onde a ordem dos $n + 1$ operandos não pode ser alterada. De acordo com a expressão apresentanda anteriormente a solução é o n-ésimo elemento do conjunto dos números de Catalan (DAVIS, 2006).

Parte II

Experimento

O experimento desenvolvido neste trabalho consiste na criação de um jogo educacional utilizando a tecnologia HTML5. Como tecnologias complementares foram utilizadas: Javascript, CSS3, Queue.js, Highcharts e PhoneGap. O conteúdo educacional escolhido para ser trabalhado no jogo foi expressões algébricas. O jogo consiste em apresentar uma expressão ao jogador e pedir para que ele selecione e resolva cada uma das operações até chegar ao resultado final.

5 Visão geral do fluxo

O fluxo tem início quando o usuário acessa o jogo – como página de internet ou aplicativo para dispositivo móvel. O jogo inicializa e apresenta o botão “Play” para que o jogador comece os testes. Quando iniciados os testes o jogo escolhe uma expressão e a apresenta ao usuário para que ele selecione uma operação para resolver.

Assim que uma operação é selecionada o programa avalia se esta pode ser resolvida e apresenta a pontuação. Caso a operação selecionada não possa ser resolvida o programa se mantém no estado de seleção, no caso contrário o programa pede a solução para a operação selecionada. Depois que o usuário passa a solução o jogo apresenta a pontuação. Se a solução passada estiver incorreta o jogo permanece no estado de resolução, se a solução estiver correta a aplicação apresenta a expressão resultante.

Enquanto houver operações não resolvidas o fluxo se repete desde a seleção de operação. Quando não houver mais operações a serem resolvidas o programa apresenta os botões “Next...” e “Exit” para o jogador fazer o próximo teste ou sair do jogo respectivamente. Se o jogador for para o próximo teste o fluxo volta para o passo de escolha de expressão, no caso do jogador sair o programa apresenta a tela final.

5.1 Inicialização da aplicação

A função start é responsável pela inicialização da aplicação, recebe como parâmetro o identificador do elemento HTML que irá conter a tela do jogo e insere os principais elementos HTML do jogo, inclusive a tela inicial para que o usuário possa iniciar o jogo.

5.2 Escolha da expressão

A escolha da expressão é feita através da função newXp. Inicialmente a rotina seleciona uma expressão, escolhendo uma das disponibilizadas através do método startDB (executado no início dos testes) ou gerando uma com a função makeExp. Em seguida a expressão selecionada é passada para o contexto do jogo junto de suas soluções.

A criação de novas expressões leva em conta duas variáveis: número de operações e um conjunto de possíveis operações a serem utilizadas. As expressões são criadas de forma aleatória. A probabilidade para cada operação ser sorteada é 1 em n, onde n é o tamanho do conjunto de operações passado. Não existe restrição de unicidade para o conjunto de operações que são passadas para a função makeExp possibilitando alterar a probabilidade de que uma determinada operação seja escolhida.

Caso a expressão escolhida tenha sido gerada a fase de passar a expressão para o contexto do jogo envolve a avaliação da expressão para conhecer todas as soluções possíveis para a expressão. A avaliação da expressão é feita através de um algoritmo iterativo implementado na função evaluateTreeIt.

5.3 Apresentação da expressão

A expressão é apresentada utilizando a função appendXp que recebe como parâmetro a expressão e a imprime na tela utilizando a função htmlfy, uma função recursiva que retorna os elementos HTML que representam visualmente a expressão.

5.4 Selecão da operação

A seleção da operação é acionada através de um clique simples com o botão esquerdo do mouse ou toque (no caso de dispositivos móveis) sobre a operação. A função opClick é responsável por tratar tais eventos recebendo como parâmetro o identificador da operação. Internamente a função opClick faz uso da função select que é responsável por avaliar se a operação pode ou não ser resolvida.

5.5 Pontuação

Sempre que houver uma seleção ou tentativa resolver uma operação o jogo avalia se a ação está correta e então apresenta a pontuação através da função spanWarn que recebe como parâmetros o tipo de pontuação, perda ou ganho, e o respectivo valor.

Quando o jogador acerta uma seleção ou resolução na primeira tentavia ele recebe 1 ponto inteiro. Para cada erro o ponto é divido ao meio, assim que se o acerto ocorrer na segunda tentativa o jogador recebe meio ponto. A seleção da última operação não é pontuada por ser uma opção única.

5.6 Pedido de solução

Sempre que uma operação for selecionada corretamente o jogo pede ao usuário a solução através da função askForSolution que apresenta ao usuário a operação isolada seguida de uma caixa de texto, para que o usuário insira a solução daquela operação. Os elementos HTML apresentados contém aqueles retornados pela função htmlfy.

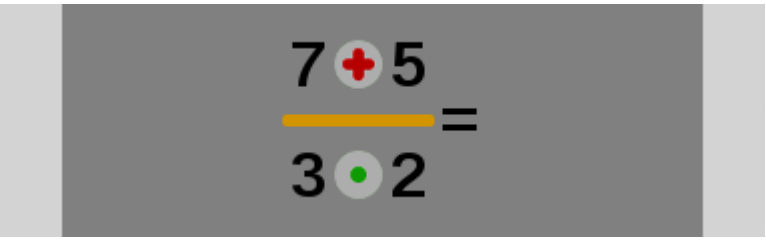
5.7 Passagem da solução

Depois que o usuário inserir a solução na caixa de texto e apertar a tecla ENTER ou tirar o foco da caixa de texto o programa faz uma avaliação da solução passada através da função solSubmit. Caso a solução esteja correta a caixa de texto é transformada em texto simples. Se houver mais de uma possível solução apenas aquela que foi resolvida permanece na tela, as outras são eliminadas da tela.

5.8 Exemplo

O exemplo a seguir exemplifica como o fluxo do jogo é percebido pelo usuário através da interface. Após inicializar a aplicação o usuário vê a tela inicial e o botão “Play”. Após apertar o botão “Play” o jogo apresenta uma expressão para o jogador resolver.

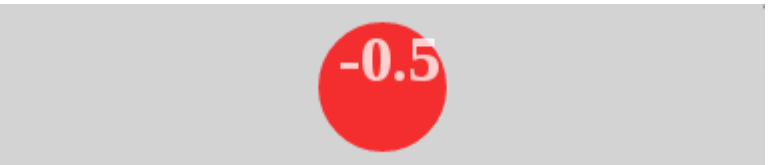
Figura 3 – O sistema apresenta a expressão a ser resolvida



Fonte: Produzido pelo autor

O jogador seleciona uma operação que não pode ser resolvida e perde metade da pontuação da seleção.

Figura 4 – Jogador perde metade da pontuação da seleção



Fonte: Produzido pelo autor

Em seguida o usuário seleciona a uma operação que pode ser resolvida e ganha a pontuação da seleção.

Figura 5 – Jogador ganha a pontuação da seleção

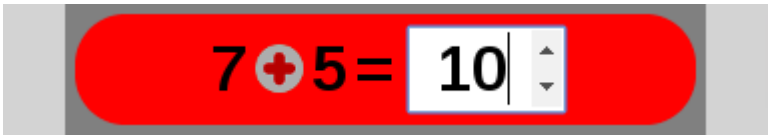
Fonte: Produzido pelo autor

O programa então apresenta a operação isolada e a caixa onde o usuário insere a solução. Este passo será omitido no resto do exemplo.

Figura 6 – Programa pede a solução da operação

O jogador erra a solução e perde metade da pontuação da resolução.

Figura 7 – O usuário insere um valor errado como solução



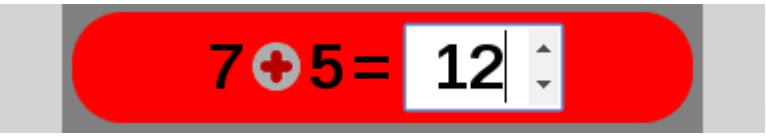
Fonte: Produzido pelo autor

Figura 8 – O jogador perde metade dos pontos da solução

Fonte: Produzido pelo autor

Em seguida o valor é corrigido e o usuário ganha a pontuação da solução.

Figura 9 – O jogador insere a solução correta



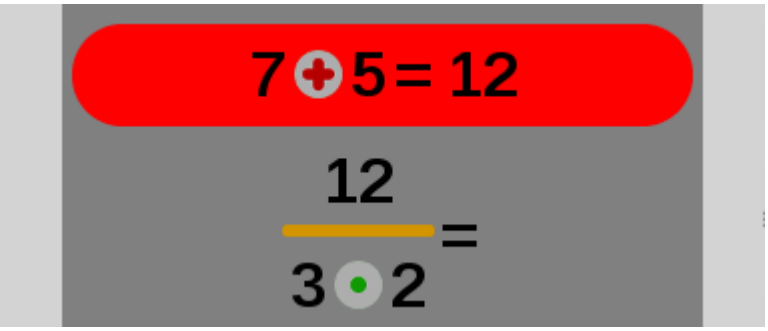
Fonte: Produzido pelo autor

Figura 10 – O sistema pontua o jogador pela solução

Fonte: Produzido pelo autor

O campo de texto é transformado em texto simples e a expressão resultante é apresentada ao usuário. O fluxo seleção/resolução se repete.

Figura 11 – O sistema mostra a expressão resultante



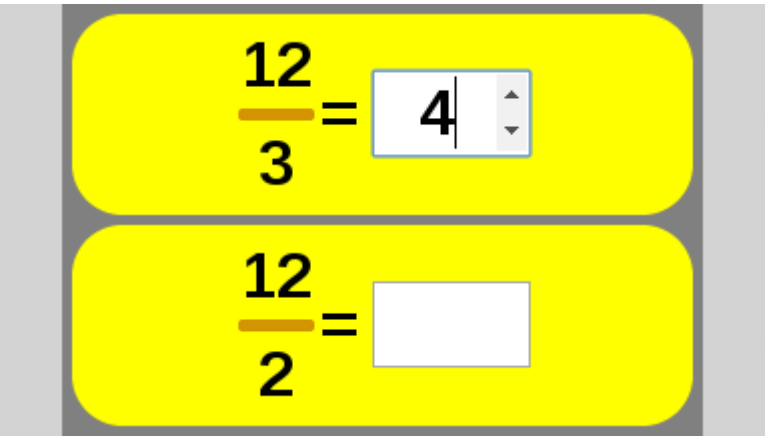
Fonte: Produzido pelo autor

Figura 12 – O jogador recebe 1 ponto pelo acerto da seleção

Fonte: Produzido pelo autor

A divisão pode ser resolvida de duas maneiras e o jogo apresenta as duas possibilidades de solução.

Figura 13 – O jogo aparesenta duas possibilidades de resolução para a divisão



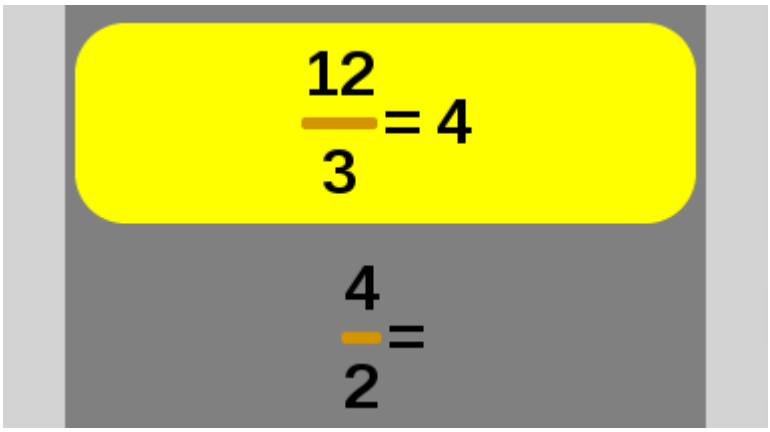
Fonte: Produzido pelo autor

Figura 14 – O jogador recebe 1 ponto pelo acerto da solução

Fonte: Produzido pelo autor

O fluxo seleção/resolução se repete.

Figura 15 – A nova expressão é apresentada ao usuário



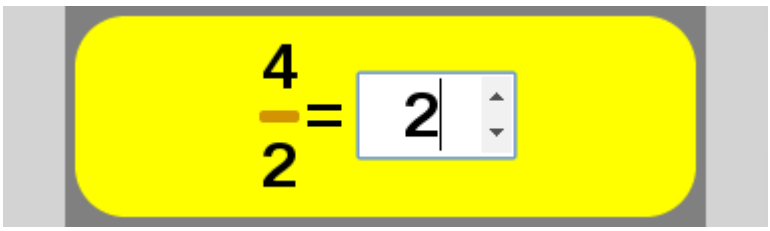
Fonte: Produzido pelo autor

A seleção da última operação não é pontuada.

Figura 16 – A seleção da última operação não é pontuada

Fonte: Produzido pelo autor

Figura 17 – O jogador insere a última solução a que está correta



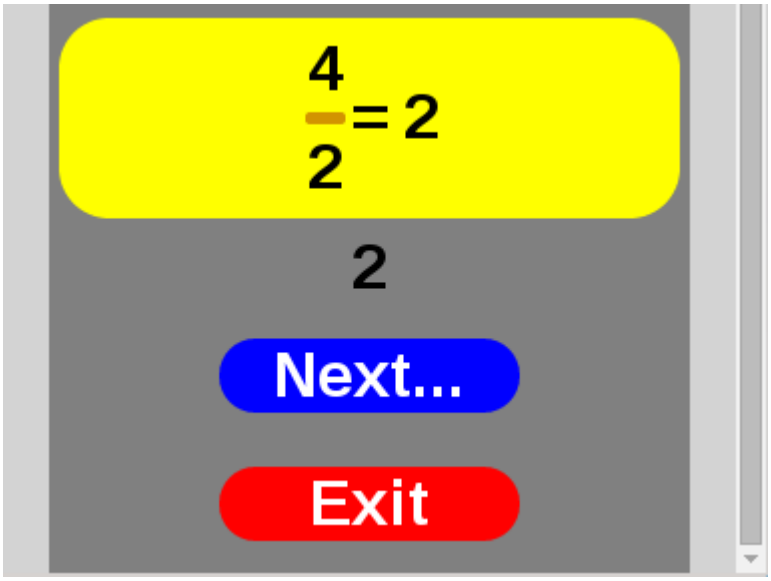
Fonte: Produzido pelo autor

Figura 18 – O jogador recebe a pontuação referente a solução

Fonte: Produzido pelo autor

Não existem mais operações para serem resolvidas e as opções de continuar ou sair são apresentadas ao usuário.

Figura 19 – A tela final é apresentada



Fonte: Produzido pelo autor

6 Criação de expressões

As expressões algébricas utilizadas são linguagens formais pois são um subconjunto de todas as palavras existentes para um determinado alfabeto. Para tanto existe pelo menos uma gramática capaz de gerar a linguagem formal envolvida.

6.0.1 A linguagem formal envolvida no jogo

Vamos partir do conjunto de símbolos terminais que deve conter: as operações de soma, subtração, multiplicação e divisão; um conunto de números dado por um intervalo fechado sobre o conjunto dos inteiros; parênteses. Logo temos T = +, -, *, /, -2, -1, 0, 1, 2, (,). Escolhemos um pequeno intervalo de inteiros neste caso para manter o conjunto pequeno.

As expressões que fazem da linguagem formal envolvida são do tipo infixa, ou seja, a operação é posicionada no meio de dois elementos que podem ser outras uma outra operação ou um inteiro. Esta escolha se dá por ser o formato que os alunos estão acostumados a ver.

6.0.1.1 Prefixo, infixo, pósfixo

Podemos representar uma operação matemática utilizando os três seguintes formatos: prefixo, infixo e pósfixo. No primeiro formato a operação é posicionada em primeiro, antes dos dois operandos. O formato pósfixo é simétrico ao formato anterior, ou seja, a operação é posicionada em último depois dos operandos. Uma vantagem da utilização dos formatos anteriores é a não necessidade de utilizar parênteses para expressar precedência entre as operações. Já no formato infixo a operação é posicionada entre os operandos e necessita utilizar parênteses para expressar precedência entre as operações.

6.0.1.2 Parênteses

A necessidade de utilizar parênteses para expressar precedência entre operações na forma infixa não ocorre para todas as operações na expressão. Os casos são os seguintes: quando uma subtração tem como operando direito outra operação que seja uma soma ou subtração; quando o operando de uma multiplicação ou divisão for uma operação de soma ou subtração; quando um número negativo é operando direito de uma operação.

6.0.1.3 Gramática referência

Tendo em conta os pontos apresentados anteriormente é possível criar uma gramática que seja capaz de gerar as expressões que serão utilizadas no jogo. Algumas

simplificações serão feitas a seguir nas produções. Os terminais o e i representam os seguintes conjuntos de terminais respecitvamente o conjunto das operações evolvidas e o conjunto expresso por um intervalo fechado e contínuo sobre os números inteiros.

Com a simplificação anterior não existe necessidade de criar uma regra de produção para cada operação e para cada inteiro. Subentende-se que estes terminais, que representam conjuntos, poderiam se desdrobrar em um símbolo não-terminal que produz cada um dos elementos do conjunto que representam sozinho.

A gramática pensada como referência para a solução adotada na criação das expressões é a seguinte:

$G : (N, T, P, S)$, onde:

$N = \{S, E\}$,

$T = \{o, i, (,)\}$,

P é o conjunto das seguintes produções:

$S \rightarrow (EoE)$

$E \rightarrow (EoE)|(i)$,

$\sigma = S$.

Note que a linguagem possui parênteses para todas as operações e inteiros. A linguagem foi pensada assim pois a forma ecolhida para representar suas palavras não foi um texto simples mas sim uma árvore que por natureza já expressa as precedências entre as operações além da necessidade dos parênteses para alguns números inteiros negativos. A lógica de apresentação de parênteses está encapsulada na função htmlfy.

De acordo com a Classificação de Chomsky, apresentada anteriormente, tal gramática é Livre de Contexto.

6.0.2 Algoritmo para criação de expressões

No código Javascript contido no arquivo xpress.js temos a função makeExp responsável pela criação das expressões. A função recebe como parâmetros o número de operações que a expressão deve conter e um conjunto contendo quais operações deversão fazer parte da expressão. A lógica da função makExp está encapsulada em duas funções, makeOps e fillWithInts, onde a primeira gera uma árvore de operações apenas e a segunda preenche a árvore com inteiros, lembrando que a árvore utilizada para representar as expressões é do tipo binária já que cada operação possui dois operandos.

6.0.2.1 makeOps

A função makeOps tem os mesmos parâmetros da função makeExp. Caso as operações não sejam passadas a função utiliza todas as operações (soma, subtração, multiplicação e divisão). A rotina começa com a criação de: uma lista de possíveis operações pai iniciada vazia; uma referência para a raíz da árvore de operações que começa com o valor nulo.

O passo seguinte é uma iteração sobre o número de operações a serem criadas. Uma operação daquelas passadas como parâmetro é sorteada e passada como parâmetro para a função criadora dos nós que retorna um nó com aquela determinada operação. Se a referência a raíz feita fora da iteração ainda possuir o valor nulo a referência apontará então para o nó criado. No caso contrário o programa sorteia um possível pai da lista, remove da mesma, definindo este como pai do nó criado.

No final da iteração o programa adiciona novas entradas na lista de possíveis pais. Estas entradas representam a ideia de que o ultimo nó criado pode ser pai a esquerda e pai a direita. Quando não existem mais iterações para acontecer a função retorna o nó referenciado como raíz.

6.0.2.2 fillWithInts

No caso de uma árvore incompleta, ou seja, existe pelo menos um operando de uma operação que não foi definido a função fillWithInts pode torná-la completa. A função recebe como parâmetro a raíz da árvore a ser preenchida com inteiros. A função faz uma busca em profundida na árvore para encontrar todas as operações que possuem operandos não definidos e então os define.

A busca em profundidade é feita utilizando uma pilha de nós a serem analisados. A pilha começa com a raíz que foi passada como parâmetro. Enquanto houver nós na pilha o programa tira o último nó da pilha testando se seus operandos, esquerdo e direito, são nós ou se estão vazios. No caso de o operando ser uma operação o programa o empilha para ser futuramente analisado, se não a rotina cria um nó com um número inteiro e o define como o operando que falta.

6.0.2.3 Restrição

Embora o algoritmo criado seja capaz de criar expressões com todas as operações (soma, subtração, multiplicação e divisão) no experimento este é utilizado somente para criar expressões com somas e subtrações, as duas com a mesma probabilidade.

A restrição foi feita para limitar o escopo deste trabalho pois como requerimento todos os números envolvidos nas expressões, inclusive aqueles que são a solução de uma operação, devem ser inteiros. Neste caso se as expressões possuem divisão o algoritmo

na expressão fillWithInts deveria levar tal fato em conta e garantir que os resultados das divisões sejam números inteiros.

6.1 Análise de soluções

As expressões algébricas envolvidas no experimento podem ter mais de uma forma de solução e para que o jogo possa avaliar corretamente a resolução da expressão feita pelo usuário é necessário fazer a avaliação da expressão a fim de encontrar todas as ordens de resolução possível.

6.1.1 Ambiguidade

A possibilidade de mais de uma solução para uma mesma expressão se dá pelo fato de que esta expressão pode ser gerada de mais de uma forma utilizando as produções. Por exemplo, 3+4+5 é uma expressão ambigua pois existem duas árvores que podem representá-la:

ARVORE

Como vemos a precedência entre as operações varia de uma árvore para a outra. Na primeira a operação 3+4 não tem nenhuma dependência e pode ser resolvida, na segunda isto ocorre com a operação 4+5. A transformação que explica a mudança da primeira árvore para a segunda é uma rotação no sentido horário feita na raíz da árvore. A possibilidade da rotação no sentido horário se dá pois não existe precedência real entre as operações envolvidas significando que as duas podem ser resolvidas no primeiro passo de resolução.

A rotação utilizada não muda o significado da expressão. Tal fato também ocorre em árvores binárias ordenadas as quais podem ter seus nós rotacionados indeterminadamente sem perder a ordenação de seus elementos. Embora exista tal semelhança as árvores que respresentam as expressões neste experimento não podem ser rotacionadas livremente sem que o significado se altere.

A subtração de uma soma é um caso onde a rotação das operações não pode ocorrer. Por exemplo, 5-(4+3). Existe uma precedência lógica nesta expressão que representa a necessidade de primeiro somar para depois subtrair. A árvore que representa tal expressão é esta:

ARVORE

A lógica que explica a utilização dos parênteses em operações é a mesma lógica que explica a possibilidade de rotação na árvore sem perder o significado, ou seja, como a soma é o operando direito da subtração esta deve estar entre parênteses e por sua vez a subtração não pode ser rotacionada no sentido. Além da lógica anterior que explica os

parênteses a rotação só pode ocorrer se o nó a ser rotacionado e aquele que toma o seu lugar são operações.

6.1.2 Rotação das árvores

Como mencionado anteriormente as árvores podem ser rotacionadas tanto no sentido horário como no sentido anti-horário. A lógica que explica a possibilidade de rotação varia entre a rotação horária e a anti-horária já que a consideração que deve ser feita em relação a subtração ocorre apenas quando tentamos fazer uma rotação anti-horária (como no caso anterior da expressão 5-(4+3)).

6.1.2.1 Rotação horária

A rotação neste sentido é sempre possível para as operações que possuem como filho esquerdo outra operação, já que a única restrição a ser considerada para saber se a rotação pode ser feita se refere apenas ao filho direito do nó a ser rotacionado.

6.1.2.2 Rotação anti-horária

A rotação anti-horária não pode ser feita para toda operação que tem como filho direito outra operação, a operação pai em questão não pode ser uma subtração.

6.1.3 Números de Catalan

Como mostrado por Robert Sedgewick e Phillipe Flajolet (2013) os números Catalãos explicam o número de possíveis árvores binárias ordenadas de acordo com o número de nós internos da árvore inicial. O teorema Enumeração de Árvores Binárias enuncia: “O número de árvores binárias com N nós internos e N+1 nós externos é dado pelos números Catalãos” (Robert Sedgewick e Phillipe Flajolet, 2013).

No caso do experimento algumas rotações não são possíveis nas expressões e sendo assim temos como valor máximo para o tamanho do conjunto das possíveis árvores o número Catalão associado ao número de nós internos na árvore. No entanto se dividirmos as árvores aonde existem as retrições de giro, subtrações com operações como filho direito, podemos encontrar o número de possíveis árvores. Depois de dividir a árvore nos pontos de restrição fazemos a análise para cada árvore resultante e multiplicamos os números para encontrar o número de possíveis árvores.

6.1.4 Algoritmo para análise de soluções

A criação de expressões é um ponto importante no experimento desenvolvido neste tabalho pois evita a necessidade da criação manual de expressões e cria um potencial

repositório infinito de expressões. Para tanto é necessário que a aplicação seja capaz de avaliar a expressão criada para encontrar todas as soluções possíveis.

O algoritmo utilizado para a análise de soluções analisa a árvore que representa a expressão dos nós mais distantes da raíz até os mais próximos. Desta forma cada nó pai conhece todas as transformações possíveis para seus nós filhos. A partir do produto cartesiano de todas as transformações possíveis o nó é testado para rotações sobre todas as transformações contidas no produto cartesiano de soluções dos filhos.

Quando o nó a ser avaliado for raíz o algoritmo testa para conhecer se as soluções avaliadas já existem em um conjunto final de soluções ou não, caso não o algoritmo adiciona tal soluções a este conjunto.

Caso uma nova solução seja encontrada esta é colocada em uma fila para que seja analisada em seguida, assim como a árvore inicial foi. O pseudo-código a seguir explica em mais detalhes o funcionamento do algoritmo para uma dereminada árvore:

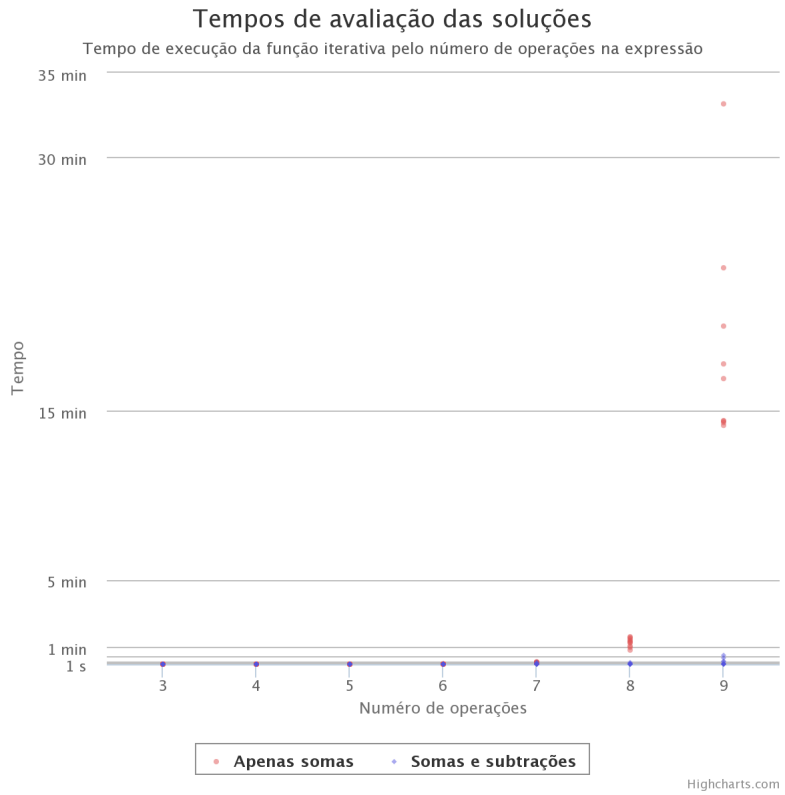
novo Mapa Soluções; adicionar árvore a Soluções; nova Fila Soluções a avaliar; enfilar árvore em Soluções a avaliar; enquanto Soluções a avaliar não é vazia nova Solução S1; tirar a primeira Solução da fila e atribuir à S1; transformar arvore utilizando as transformações de S1; nova Lista Nós a avaliar; criar Lista de nós para a raíz de S1 e atribuir à Nós a avaliar; novo Mapa Soluções por nó; para cada Nó N em Nós a avaliar Soluções por nó recebe uma nova Lista em N; nova Lista Pré-soluções; combinar soluções dos filhos direito e esquerdo de N e atribuir a Pré-soluções; para cada Solução P em Pré-soluções nova Solução S2 recebe a combinação de S1 e P; adicionar S2 à Lista no Soluções por nó em N; se N é raiz e a chave de S2 não está em Soluções adicionar S2 a Soluções e Soluções a avaliar; novo Nó H recebe N; nova Solução SH recebe S2; enaquanto H pode ser rotacionado no sentido horário rotacionar H no sentido horário; adicionar transformação feita às transformações de SH; adicionar SH à Lista no Soluções por nó em N; se o pai de H é raiz e a chave de SH não está em Soluções adicionar SH a Soluções e Soluções a avaliar; atribuir o pai de H a H; desfazer todas as rotações feitas no sentido horário; novo Nó AH recebe N; nova Solução SAH recebe S2; enaquanto AH pode ser rotacionado no sentido anti-horário rotacionar AH no sentido anti-horário; adicionar transformação feita às transformações de SAH; adicionar SAH à Lista no Soluções por nó em N; se o pai de AH é raiz e a chave de SAH não está em Soluções adicionar SAH a Soluções e Soluções a avaliar; atribuir o pai de AH a AH; desfazer todas as rotações feitas no sentido anti-horário; desfazer as transformações de P; desfazer as trasnformações de S1;

6.1.5 Desempenho do algoritmo

Para testar o desempenho do algoritmo de avaliação de expressões a função evaluateTreeItTest foi criada. A função recebe quatro parâmetros de teste. Os dois primeiros são os limites de um intervalo de números inteiro que representam os tamanhos de árvores a serem testadas. O parâmetro seguinte se refere ao número de testes por tamanho de árvore. O ultimo parâmetro é opcional, uma lista de possíveis operações a serem utilizadas na criação das expressões que serão testadas.

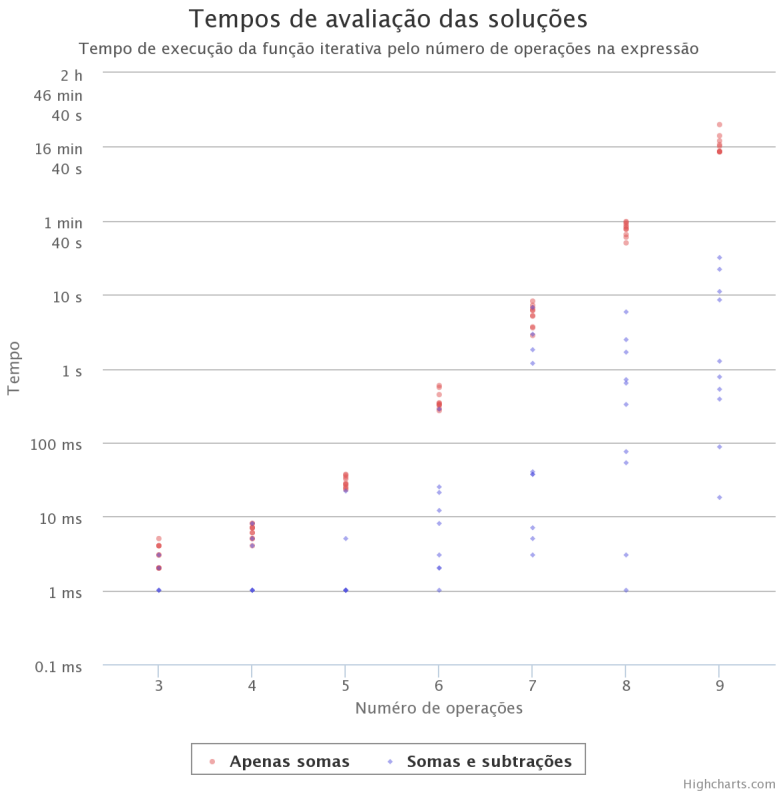
Nos gráficos a seguir podemos observar os resultados dos testes. Os teste foram feitos 10 vezes para árvores de 3 a 9 nós, onde no primeiro teste somente operações de soma foram utilizadas diferente so segundo que também utiliza subtrações.

Figura 20 – Diagrama sobre rotação de árvores binárias ordenadas



Fonte: Produzido pelo autor

Figura 21 – Diagrama sobre rotação de árvores binárias ordenadas



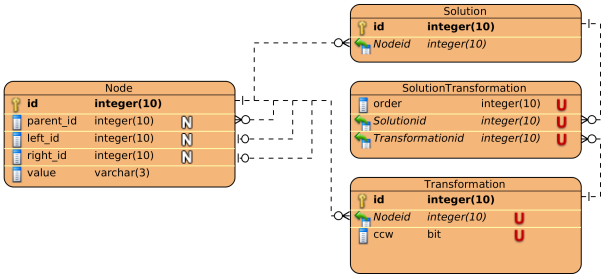
Fonte: Produzido pelo autor

A diferença nos tempos das séries sem e com soma se devem ao fato de que expressões com subtração tem limitações de rotação. O primeiro gráfico em escala linear mostra o comportamento exponencial da expressão. No gráfico cuja a escala é logarítmica podemos observar com maior clareza como as tempos nas expressões que envolvem subtração variam em um intervalo cujo tempo máximo é o tempo para as árvores que não envolvem subtrações. Isto se deve ao caráter aleatório do algoritmo de geração de expressões. Uma expressão só com somas pode ser gerada mesmo no teste que envolve também subtrações.

6.2 Modelo e estruturas de dado

Por mais que o experimento não utilize uma aplicação de banco de dados existe um modelo de dados intrínseco expresso no diagrama a seguir.

Figura 22 – Diagrama sobre rotação de árvores binárias ordenadas



Fonte: Produzido pelo autor

O diagrama representa as expressões e suas respectivas soluções. Como a estrutura de dados escolhida para representar as expressões foi a árvore, as expressões são armazenadas podem ser armazenadas da mesma forma. Cada nó (Node) aponta para o seu nó pai e seus nós filhos.

As soluções (Solution) para uma determinada expressão são armazenadas na forma de uma lista ordenada de transformações (Transformation). A lista de ordenada de transformações é representada através da entidade SolutionTransformation.

Parte III

Tecnologias

7 HTML5

De acordo com o texto de introdução de W3SCHOOLS HTML5 é o padrão HTML mais recente. A versão anterior (4.01) foi lançada em 1999 e como a internet mudou significantemente desde então a versão 5 vem para substituir a versão 4.01, além do XHTML e o HTML DOM Level 2. A nova versão provê de animações a gráficos, musicas a filmes além de possibilitar aplicações web complexas. O padrão HTML5 é multiplataforma e é feito para funcionar em PCs, Tablets, Smartphones e Smart TVs. O versão 5 do padrão HTML é uma cooperação entre o World Wide Web Consortium (W3C) e o Web Hypertext Application Tecnology Working Group (WHATWG). Ambas as organizações trabalhavam em diferentes aspectos de aplicações web e em 2006 resolveram se juntar para criar a nova versão do HTML. Algumas diretrizes foram estabelecidas para o futuro empreendimento (W3SCHOOLS,):

a) Novas funcionalidades devem estar baseadas em HTML, CSS, DOM e JavaScript;

b) A necessidade de plugins externos deve ser reduzida;

c) Lidar com erros deve ser mais fácil que nas versões anteriores;

d) “Scripting” deve ser substituido por mais marcações;

e) O padrão deve ser independente de dispositivo;

f) O publico deve ter visibilidade do processo de desenvolvimento

Entre as novas “features” disponibilizadas na nova versão temos:

a) O elemento canvas para desenho em 2D;

b) Os elementos video e audio para execução de mídias;

c) Suporte para armazenamento local

7.1 O surgimento do HTML

A World Wide Web teve inicio no CERN, Laboratório para Física de Particulas em Geneva, Suiça. O conceito do HTML surgiu enquanto Tim Berners-Lee trabalhava em uma sessão de serviços computacionais do CERN. Tim teve a ideia de permitir que pesquisadores em diferentes lugares do mundo pudessem organizar e juntar informações remotamente, já que as pesquisas em Física de Particulas envolve com frequência diversos institutos de varios lugares do mundo. A questão não era apenas disponibilizar um grande número de pesquisas mas permitir que ligações entre os documentos fossem estabelecidas. Antes de trabalhar no CERN Tim havia trabalhado com produção de documentos e

processamento de texto. Tim pensou que a solução seria desenvolvida através de uma forma de hipertexto.

O protótipo de navegador web estava pronto em 1990 para o computador NeXT e foi desenvolvido por Tim Berners-Lee. O surgimento da web no começo dos anos 90 está relacionado aos desenvolvimentos em tecnologias de comunicação durante o período assim como o hipertexto ganhava espaço e comçava a ser usado em computadores. O sistema de nomes de domínios (DNS) também foi importante no surgimento da Web pois facilitava o acesso as máquinas específicas.

Tim Berners-Lee via a viabilidade dos links globais de hipertexto. Uma consideração importante feita por Tim foi a importância de que tal hipertexto deveria funcionar nos mais diversos computadores que estavam ligados à Internet. Tim desenvolve um software e um protocolo (HTTP) que eram capazes de recuperar documentos de texto através de links de hipertexto para demonstrar sua forma de publicar documentos. O texto que era utilizado pelo protocolo HTTP (HyperText Transfer Protocol) foi nomeado de HTML que significa linguagem de marcação de hipertexto (HyperText Mark-up Language).

A linguagem HTML criada por Tim Berners-Lee se baseou em um método de marcação de textos que organizava o texto em unidades estruturais como parágrafos, cabeçalhos entre outras, o SGML que era um método acordado internacionalmente. SGML significa linguagem padrão generalizada de marcação (SGML). Tal embasamento foi um fator que colaborou para o sucesso da ideia de Tim (LONGMAN, 1998). A World Wide Web teve inicio no CERN, Laboratório para Física de Particulas em Geneva, Suiça. O conceito do HTML surgiu enquanto Tim Berners-Lee trabalhava em uma sessão de serviços computacionais do CERN. Tim teve a ideia de permitir que pesquisadores em diferentes lugares do mundo pudessem organizar e juntar informações remotamente, já que as pesquisas em Física de Particulas envolve com frequência diversos institutos de varios lugares do mundo. A questão não era apenas disponibilizar um grande número de pesquisas mas permitir que ligações entre os documentos fossem estabelecidas. Antes de trabalhar no CERN Tim havia trabalhado com produção de documentos e processamento de texto. Tim pensou que a solução seria desenvolvida através de uma forma de hipertexto.

O protótipo de navegador web estava pronto em 1990 para o computador NeXT e foi desenvolvido por Tim Berners-Lee. O surgimento da web no começo dos anos 90 está relacionado aos desenvolvimentos em tecnologias de comunicação durante o período assim como o hipertexto ganhava espaço e comçava a ser usado em computadores. O sistema de nomes de domínios (DNS) também foi importante no surgimento da Web pois facilitava o acesso as máquinas específicas.

Tim Berners-Lee via a viabilidade dos links globais de hipertexto. Uma consideração importante feita por Tim foi a importância de que tal hipertexto deveria funcionar nos mais diversos computadores que estavam ligados à Internet. Tim desenvolve um software

e um protocolo (HTTP) que eram capazes de recuperar documentos de texto através de links de hipertexto para demonstrar sua forma de publicar documentos. O texto que era utilizado pelo protocolo HTTP (HyperText Transfer Protocol) foi nomeado de HTML que significa linguagem de marcação de hipertexto (HyperText Mark-up Language).

A linguagem HTML criada por Tim Berners-Lee se baseou em um método de marcação de textos que organizava o texto em unidades estruturais como parágrafos, cabeçalhos entre outras, o SGML que era um método acordado internacionalmente. SGML significa linguagem padrão generalizada de marcação (SGML). Tal embasamento foi um fator que colaborou para o sucesso da ideia de Tim. (LONGMAN, 1998)

7.2 A evlução do padrão HTML

O HTML 2, nomeado assim por Tim Berners-Lee, surgiu para resolver o problema a linguagem que estava se tornando mal-definida conforme diversos navegadores adicionavam novos elementos HTML de uma forma não coordenada. Para tanto Dan Connolly e seus colegas fizeram o trabalho de unir as diferentes variantes e colocar em um documento de rascunho. Feito o rascunho este foi circulado pela Internet para que a comunidade pudesse comentar. O resultado de tal esforço foi a especificação da nova versão do HTML.

A dificuldade em manter o padrão não foi resolvida de fato na versão 2. O HTML 3, que foi publicado como um rascunho em 1995, também sofreu a dificuldade de manter o padrão da linguagem. O rascunho muito longo teve dificuldades em ser ratificado pelo IETF devido ao tamanho do esforço estimado. O problema em manter o padrão era semelhante ao enfrentado antes da especificação, cada navegador implementava diferentes subconjuntos do padrão além de possíveis extensões à linguagem. O HTML 3 não se tornou de fato um padrão.

Por mais que o HTML 3 não se tornou padrão o rascunho fez progressos importantes lidando com tabelas, notas de rodapé, formulários além de incluir o elemento STYLE e o atributo CLASS para encorajar os autores utilizarem estilo em seus documentos. Ainda em 1995 foi apresentado o artigo sobre a internacionalização da Web que previa acabar com a restrição de cunjunto de caracteres para que outros além do Latin-1 pudessem ser utilizados.

A especificação do HTML 3.2 foi endossada em janeiro de 1997 pelo W3C (World Wide Web Consortium) e trazia como um dos novos elementos disponíveis o OBJECT para incorporar objetos, por exemplo applets, dentro de um documento HTML. O HTML 3.2 foi resultado de uma combinação de todas as especificações anteriores e foi amplamente aprovada. (LONGMAN, 1998)

A versão anterior a especificação 5 é o HTML 4.01, revisão do HTML 4. Entre as

diversas melhorias temos mecanismos para folhas de estilo, scripting e quadros. A versão 4 foi desenvolvida com ajuda de experts em internacionalização. A incorporação do RFC2070 realizou a internacionalização do HTML (W3C,).

8 Javascript

Javascript é uma linguagem de programação desenvolvida para aplicações web. A execução da linguagem ocorre no navegador web que executa a página contenedora

9 CSS3

De acordo com w3schools CSS3 é o padrão CSS mais recente que é compatível com as versões anteriores.

10 Highcharts

Highcharts é uma API Javascript utilizada para gerar gráficos em páginas HTML. Os gráficos possuem varias opções de configuração o que os tornam muito flexíveis. A biblioteca Javascript é capaz de gerar diversos tipos de gráficos como: linha, área, coluna, pizza, dispersão, relógio. A API é bastante documentada além de existirem diversos exemplos no site da produto.

11 PhoneGap

PhoneGap é um framework para desenvolvimento de aplicações mobile. A aplicação é desenvolvida utilizando HTML5, CSS3 e Javascript e encapsulada na estrutura do framework. Utilizando uma ferramenta de construção a aplicação é empacotada para celulares. O framework é multiplataforma e permite que a apliação seja empacotada para diversos dispositivos como Blackberry, Android, iPhone entre outros. A aplicação acessa os recursos do dispositivo móvel através de bibliotecas Javascript.

12 Adobe PhoneGap Build

A ferramenta Adobe PhoneGap Build é um serviço na núvem para a compilação de aplicações que utilizam o framework PhoneGap. A ferramenta permite que o usuário envie o código-fonte da aplicação e em seguida faça o download da mesma já empacotada. A utilização desta ferramenta evita que o usuário tenha que instalar os SDKs nativos de cada plataforma além do Cordova/PhoneGap SDK, necessários para a compilação da aplicação localmente.

13 Git

Git é um programa para gerenciamento de código-fonte. Versionamento é uma das principais funcionalidades de tal programa. Uma diferença que pesa a favor do Git em relação a outros programas de gerenciamento de código-fonte é a possibilidade de saber se arquivos mudam de nome e/ou diretório devido a forma como este funciona. Em outros programas tal informação se deturpa no sentido de que esta é quebrada em duas informações, o arquivo origem é tido como deletado enquanto o arquivo destino é visto como um novo arquivo criado.

14 GitHub

GitHub é um serviço de repositório para código-fonte que tem como base o programa de controle de versão Git. Existem tanto versões gratis quanto versões pagas do serviço. No caso de repositórios abertos não existe a necessidade de pagamento. O código-fonte do experimento encontra-se em github.com/camiloperic/xpress.

Parte IV

Considerações Finais

O experimento desenvolvido neste trabalho cumpre o objetivo proposto de desenvolver um protótipo de jogo educativo de expressões algébricas. O protótipo é capaz de gerar e avaliar expressões com somas e subtrações, além de permitir a solução das expressões pelo usuário. O protótipo também conta com expressões com as quatro operações em sua base de dados.

Existem diversas possibilidades de continuação deste trabalho, essas possibilidades surgem tanto de aspectos incompletos assim como aspectos que podem ser aperfeiçoados. Como apontado no texto sobre as motivações envolvidas neste trabalho existe um grande potencial na utilização de informações que podem ser geradas pelo sistema. O algoritmo de avaliação de soluções além de ter um desempenho fraco para árvores grandes este funciona apenas para somas e subtrações.

Uma possibilidade de continuação deste trabalho é adicionar um banco de dados ao sistema e estudar que tipos de análise podem ser feitas sobre as informações geradas e o que se pode aprender com estas possíveis análises.

A melhoria do desempenho do algoritmo de avaliação de soluções de expressões também pode ser objeto de um trabalho que se extende deste. Além do desempenho do algoritmo, o escopo das expressões que podem ser avaliadas pode ser expandido para incluir multiplicações e divisões.

Outro problema interessante que surgiu durante o trabalho mas não foi resolvido foi como criar expressões algébricas com somas, subtrações, multiplicações e divisões cujos números presentes sejam sempre inteiros.

Outro desdobramento possível para este trabalho é incluir novas “operações” como potenciação e radiciação, desta forma o programa pode abordar as propriedades de potenciação e radiciação por exemplo.

O código-fonte do jogo está disponível para qualquer interessado através do serviço de repositórios GitHub em http://github.com/camiloperic/xpress.

Referências

ARTALE, A. *Formal languages and Compilers - Lecture II: Formal Language Theory*. 2014. Disponível em: <http://www.inf.unibz.it/~artale/Compiler/slide2.pdf>. Acesso em: 12 ago. 2014. Citado na página 27.

DAVIS, T. *Catalan Numbers*. 2006. Disponível em: <http://mathcircle.berkeley.edu/BMC6/pdf0607/catalan.pdf>. Acesso em: 11 ago. 2014. Citado na página 33.

GLADKII, A. V. *Formal language*. Disponível em: <http://www.encyclopediaofmath.org/index.php?title=Formal_language&oldid=18696>. Acesso em: 11 ago. 2014. Citado na página 27.

LERMA, M. A. *Languages and Grammars*. Disponível em: <http://www.math.northwestern.edu/~mlerma/courses/cs310-04w/notes/dm-grammars.pdf>. Acesso em: 25 jun. 2014. Citado na página 28.

LONGMAN, A. W. *A history of HTML*. 1998. Disponível em: <http://www.w3.org/People/Raggett/book4/ch02.html>. Acesso em: 25 jun. 2014. Citado 2 vezes nas páginas 60 e 61.

SLEATOR, D. D.; TARJAN, R. E.; THURSTON, W. P. *Rotation distance, triangulations and hyperbolic geometry*. 1988. Disponível em: <http://www.ams.org/journals/jams/1988-01-03/S0894-0347-1988-0928904-4/S0894-0347-1988-0928904-4.pdf>. Acesso em: 12 ago. 2014. Citado na página 29.

STANLEY, R.; WEISSTEIN, E. W. *"Catalan Number"De MathWorld–A Wolfram Web Resource*. Disponível em: <http://mathworld.wolfram.com/CatalanNumber.html>. Acesso em: 11 ago. 2014. Citado na página 33.

W3C. *Introduction to HTML4*. Disponível em: <http://www.w3.org/TR/REC-html40/intro/intro.html>. Acesso em: 25 jun. 2014. Citado na página 62.

W3SCHOOLS. *HTML5 Introduction*. Disponível em: <http://www.w3schools.com/html/html5_intro.asp>. Acesso em: 25 jun. 2014. Citado na página 59.

Apêndices

APÊNDICE A – Quisque libero justo

Quisque facilisis auctor sapien. Pellentesque gravida hendrerit lectus. Mauris rutrum sodales sapien. Fusce hendrerit sem vel lorem. Integer pellentesque massa vel augue. Integer elit tortor, feugiat quis, sagittis et, ornare non, lacus. Vestibulum posuere pellentesque eros. Quisque venenatis ipsum dictum nulla. Aliquam quis quam non metus eleifend interdum. Nam eget sapien ac mauris malesuada adipiscing. Etiam eleifend neque sed quam. Nulla facilisi. Proin a ligula. Sed id dui eu nibh egestas tincidunt. Suspendisse arcu.

APÊNDICE B – Nullam elementum urna vel imperdiet sodales elit ipsum pharetra ligula ac pretium ante justo a nulla curabitur tristique arcu eu metus

Nunc velit. Nullam elit sapien, eleifend eu, commodo nec, semper sit amet, elit. Nulla lectus risus, condimentum ut, laoreet eget, viverra nec, odio. Proin lobortis. Curabitur dictum arcu vel wisi. Cras id nulla venenatis tortor congue ultrices. Pellentesque eget pede. Sed eleifend sagittis elit. Nam sed tellus sit amet lectus ullamcorper tristique. Mauris enim sem, tristique eu, accumsan at, scelerisque vulputate, neque. Quisque lacus. Donec et ipsum sit amet elit nonummy aliquet. Sed viverra nisl at sem. Nam diam. Mauris ut dolor. Curabitur ornare tortor cursus velit.

Morbi tincidunt posuere arcu. Cras venenatis est vitae dolor. Vivamus scelerisque semper mi. Donec ipsum arcu, consequat scelerisque, viverra id, dictum at, metus. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut pede sem, tempus ut, porttitor bibendum, molestie eu, elit. Suspendisse potenti. Sed id lectus sit amet purus faucibus vehicula. Praesent sed sem non dui pharetra interdum. Nam viverra ultrices magna.

Aenean laoreet aliquam orci. Nunc interdum elementum urna. Quisque erat. Nullam tempor neque. Maecenas velit nibh, scelerisque a, consequat ut, viverra in, enim. Duis magna. Donec odio neque, tristique et, tincidunt eu, rhoncus ac, nunc. Mauris malesuada malesuada elit. Etiam lacus mauris, pretium vel, blandit in, ultricies id, libero. Phasellus bibendum erat ut diam. In congue imperdiet lectus.

Anexos

ANEXO A – Morbi ultrices rutrum lorem.

Sed mattis, erat sit amet gravida malesuada, elit augue egestas diam, tempus scelerisque nunc nisl vitae libero. Sed consequat feugiat massa. Nunc porta, eros in eleifend varius, erat leo rutrum dui, non convallis lectus orci ut nibh. Sed lorem massa, nonummy quis, egestas id, condimentum at, nisl. Maecenas at nibh. Aliquam et augue at nunc pellentesque ullamcorper. Duis nisl nibh, laoreet suscipit, convallis ut, rutrum id, enim. Phasellus odio. Nulla nulla elit, molestie non, scelerisque at, vestibulum eu, nulla. Ut odio nisl, facilisis id, mollis et, scelerisque nec, enim. Aenean sem leo, pellentesque sit amet, scelerisque sit amet, vehicula pellentesque, sapien.

ANEXO B – Cras non urna sed feugiat cum sociis natoque penatibus et magnis dis parturient montes nascetur ridiculus mus

Sed consequat tellus et tortor. Ut tempor laoreet quam. Nullam id wisi a libero tristique semper. Nullam nisl massa, rutrum ut, egestas semper, mollis id, leo. Nulla ac massa eu risus blandit mattis. Mauris ut nunc. In hac habitasse platea dictumst. Aliquam eget tortor. Quisque dapibus pede in erat. Nunc enim. In dui nulla, commodo at, consectetuer nec, malesuada nec, elit. Aliquam ornare tellus eu urna. Sed nec metus. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

ANEXO C – Fusce facilisis lacinia dui

Phasellus id magna. Duis malesuada interdum arcu. Integer metus. Morbi pulvinar pellentesque mi. Suspendisse sed est eu magna molestie egestas. Quisque mi lorem, pulvinar eget, egestas quis, luctus at, ante. Proin auctor vehicula purus. Fusce ac nisl aliquam ante hendrerit pellentesque. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Morbi wisi. Etiam arcu mauris, facilisis sed, eleifend non, nonummy ut, pede. Cras ut lacus tempor metus mollis placerat. Vivamus eu tortor vel metus interdum malesuada.