Camilo Peric de Freitas

Elaboração de um jogo para computador envolvendo expressões algébricas

São Paulo, SP

2014

Camilo Peric de Freitas

Elaboração de um jogo para computador envolvendo 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 para computador envolvendo expressões algébricas/ Camilo Peric de Freitas. – São Paulo, SP, 2014-
79 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. Tecnologias da Informação. 2. Educação. I. Marcelo Novaes de Rezende. II. Escola Superior de Engenharia e Gestão. IV. Elaboração de um jogo para computador envolvendo expressões algébricas

Camilo Peric de Freitas

Elaboração de um jogo para computador envolvendo expressões algébricas

Trabalho aprovado. São Paulo, SP, 25 de junho de 2014:

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. O processo de graduação acaba mas tudo o que foi construído durante será levado 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.

"Imagination is more important than knowledge. (Albert Einstein)

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

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

Key-words: Information Technologies. 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, que tem como principais requisitos a criação e a avaliação de expressões algébricas.

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 envolver 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 utilizando 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. Um sistema capaz de gerar problemas educacionais tem um conjunto infinito de problemas em potencial e não requer um recurso humano para a produção de problemas.

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 Educação em 2008 foi 972.826. As instituições credenciadas incluem desde

ensino fundamental até pós-graduação. (ABRAED,)

A plataforma online de ensino Code School[1] que ensina diversas habilidades em programação e web design já implementa cursos[2] 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.

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

O trabalho foi divido em três seções nesta ordem: Referencial Teórico, Tecnologias e Experimento. Na primeira seção são apresentados os referenciais teóricos referentes aos conteúdos envolvidos nos principais problemas apresentados no experimento. A seção Tecnologias apresenta as tecnologias utilizadas no desenvolvimento da aplicação do experimento. Experimento, a última seção, explica o experimento desenvolvido neste trabalho.

Em Referencial Teórico é explicada a teoria relativa aos principais conteúdos envolvidos nos dois grandes problemas explorados no experimento. A criação das expressões algébricas foi baseada no conhecimento de linguagens e gramáticas. As propriedades das operações explicam porque pode existir mais de uma ordem de resolução para cada expressão algébrica. Algumas características do conjunto de ordens de resolução são esclarecidas por meio do conhecimento sobre árvores binárias - que envolve os números de Catalan.

Na seção Tecnologias um conjunto de tecnologias são apresentadas. A tecnologia HTML5 é abordada separadamente e em mais profundidade por ser a principal tecnologia envolvida - as outras tecnologias ou são baseadas em HTML5 ou são utilizadas como complementares (e.g. controle de versão).

A última parte do trabalho, Experimento, explica o funcionamento do protótipo apresentado. Inicialmente um modelo de dados é proposto baseado na representação das

[1] Disponível em: <https://www.codeschool.com/>. Acessado em 25, jun. 2014

[2] O curso "Shpaing up with Angular.JS" é um exemplo de tais cursos e por ser gratuito pode ser acessado sem a necessidade de inscrição na plataforma. Disponível em: <https://www.codeschool.com/courses/shaping-up-with-angular-js/>. Acessado em: 25, jun. 2014.

expressões como árvores binárias. Em seguida os dois principais problemas solucionados neste experimento são apresentados e explicados juntos das soluções encontradas.

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, 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 por meio 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

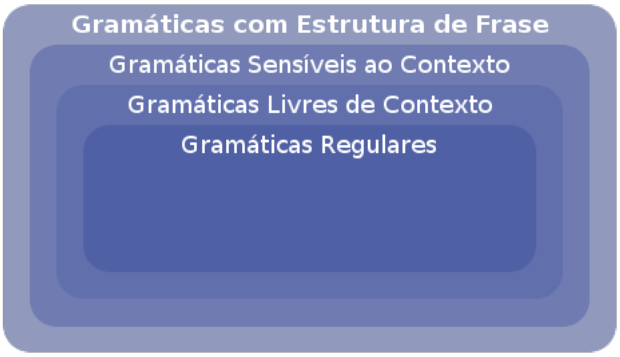
1 Linguagens e gramáticas

Uma linguagem é um conjunto arbitrário 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 Operações no Conjunto dos Números Inteiros

O Conjunto dos Números Inteiros é fechado sob as operações de soma, subtração e multiplicação, isto significa que para qualquer uma destas operações o resultado é um número inteiro caso os operandos também sejam, mas não somente. A tabela a seguir apresenta exemplos para as propriedades comutativa e associativa das operações de soma e multiplicação.

Tabela 1 – Propriedades associativa e comutativa da soma e multiplicação sobre os Números Inteiros.

Propriedade comutativa (adição/subtração)	$a + b = b + a$	$ab = ba$
Propriedade associativa (adição/subtração)	$(a + b) + c = a + (b + c)$	$(ab)c = a(bc)$

Fonte: Produzido pelo autor.

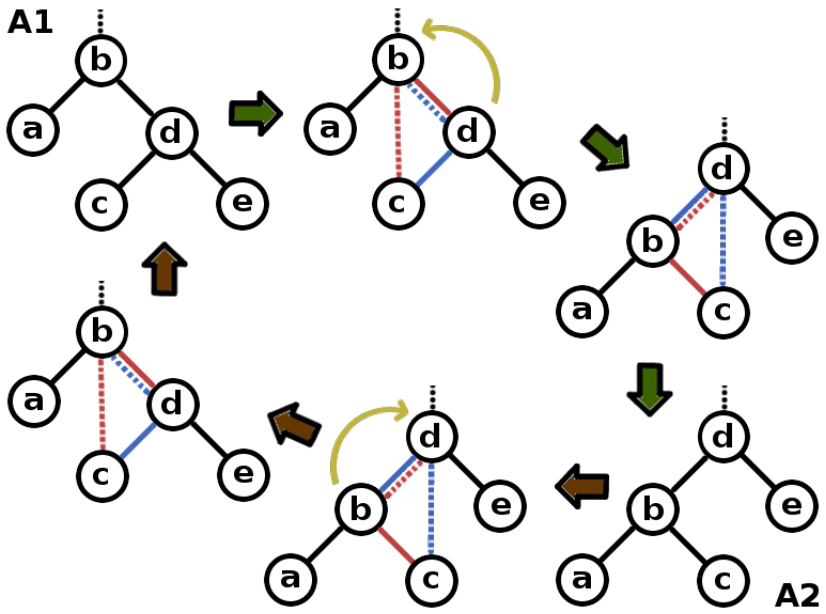
A propriedade comutativa garante que a alteração da ordem dos operandos da operação não altera o seu resultado. A propriedade associativa diz respeito à mais de uma operação e garante que a ordem em que as operações são resolvidas não altera o resultado final das operações combinadas - na tabela apresentada anteriormente os parênteses não são necessários já que não existe precedência entre as operações (característica da operação associativa) e estão presentes apenas para explicitar a ordem de resolução.

3 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. Os nós externos não possuem nós filhos, ao contrário dos nós internos que possuem dois nós filhos, um à esquerda e o outro da direita.

Árvores binárias são utilizadas em aplicações relacionadas a computador para guardar uma coleção de informações de forma ordenada, a ordem das informações é representada pela ordem simétrica dos nós. A ordem simétrica dos nós pode ser obtida utilizando o seguinte algoritmo recursivo: Se o nó é um nó interno passe pela subárvore à esquerda em ordem simétrica, passe pelo próprio nó e em seguida passe pela subárvore à direita em ordem simétrica. Se o nó é externo apenas passe por ele. A ordem de passagem dos nós é chamda de permutação de ordem simétrica dos nós ou apenas ordem simétrica dos nós. (SLEATOR; TARJAN; THURSTON, 1988)

Figura 2 – Diagrama sobre rotação de árvores binárias ordenadas



Fonte: Produzido pelo autor

A rotação é uma operação que transforma uma árvore binária em outra mantendo a ordem simétrica dos nós. Em uma árvore de tamanho n existem $n - 1$ rotações possíveis (SLEATOR; TARJAN; THURSTON, 1988). O diagrama contido na Figura 3 mostra graficamente como as rotações anti-horária e horária acontecem. A rotação anti-horária

é explicada no fluxo por meio 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 pelas 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.

4 Números de Catalan

Os números de Catalan aparecem em probelmas de enumeração de árvores como no problema de divisão dos polígonos de Euler. O n-ésimo número de Catalan pode ser obtido a partir 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

Tecnologias

5 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

5.1 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 como 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 utilizando 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).

5.2 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,).

6 Tecnologias complementares

6.1 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.

6.2 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 utilizando bibliotecas Javascript.

6.3 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.

6.4 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.

6.5 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.

6.6 Queue.js

Queue.js[1] é uma biblioteca javascript que implementa a estrutura de dado fila.

[1] Disnponível em <https://code.stephenmorley.org/javascript/queues>

Parte III

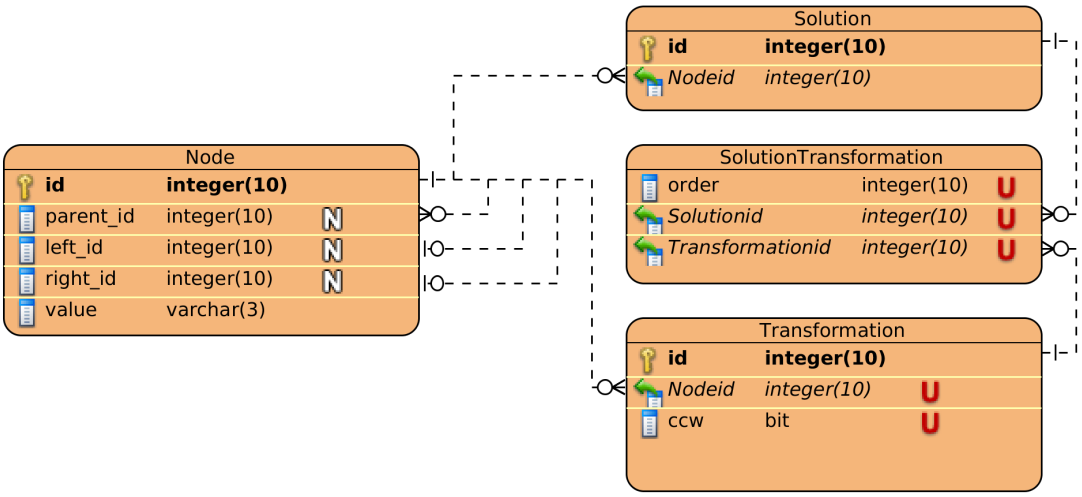
Experimento

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.

7 Modelo de dados

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 3 – Modelo de Dados



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 ordenada de transformações é representada pela entidade SolutionTransformation.

8 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.

8.1 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.

8.2 Escolha da expressão

A escolha da expressão é feita por meio da função newXp. Inicialmente a rotina seleciona uma expressão, escolhendo uma das disponibilizadas pelo 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 utilizando um algoritmo iterativo implementado pela função evaluateTreeIt.

8.3 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.

8.4 Selecão da operação

A seleção da operação é acionada por 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.

8.5 Pontuação

Sempre que houver uma seleção ou tentativa resolver uma operação o jogo avalia se a ação está correta e então apresenta a pontuação por meio 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.

8.6 Pedido de solução

Sempre que uma operação for selecionada corretamente o jogo pede ao usuário a solução utilizando a 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.

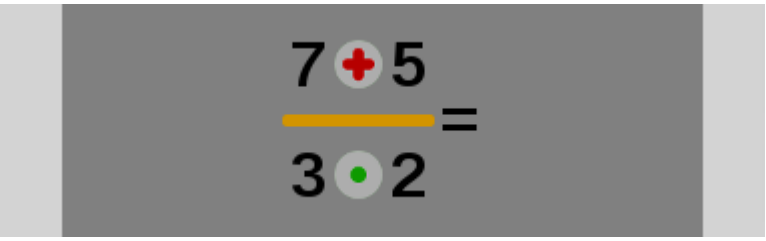
8.7 Passagem da solução

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 pela 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.

8.8 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.

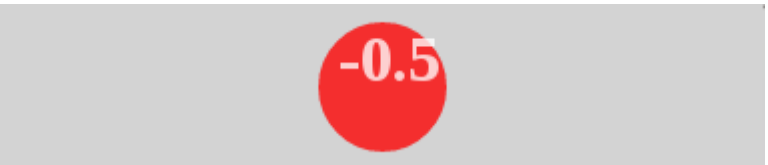
Figura 4 – O sistema apresenta a expressão a ser resolvida



Fonte: Produzido pelo autor

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

Figura 5 – 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 6 – Jogador ganha a pontuação da seleção

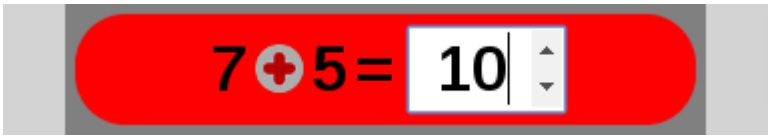
Fonte: Produzido pelo autor

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 7 – Programa pede a solução da operação

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

Figura 8 – O usuário insere um valor errado como solução



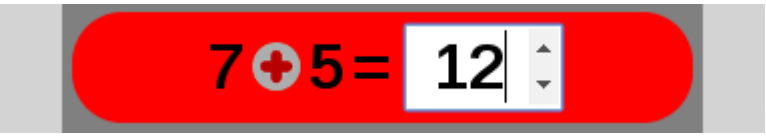
Fonte: Produzido pelo autor

Figura 9 – 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 10 – O jogador insere a solução correta



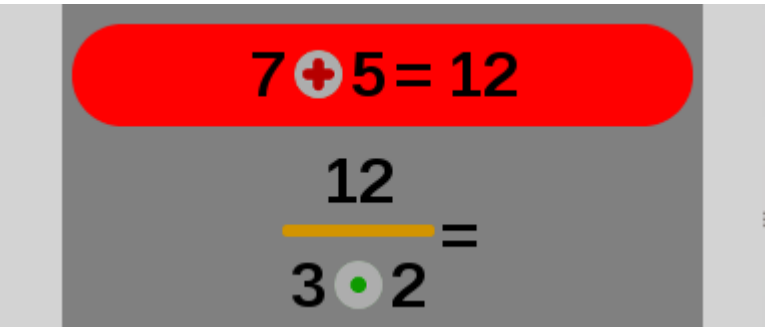
Fonte: Produzido pelo autor

Figura 11 – 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 12 – O sistema mostra a expressão resultante



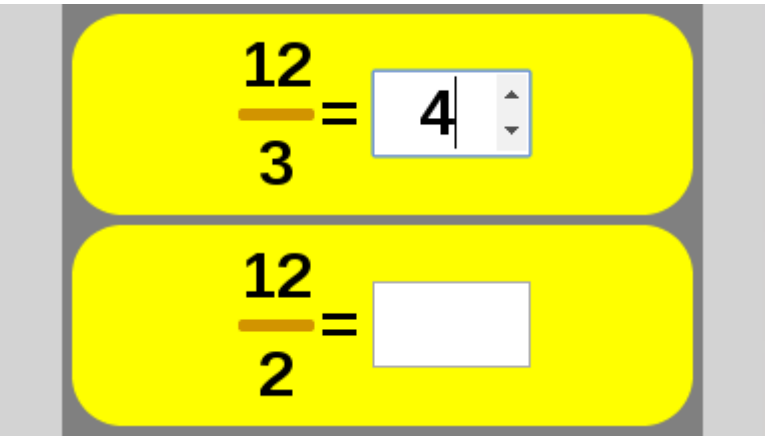
Fonte: Produzido pelo autor

Figura 13 – 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 14 – O jogo aparesenta duas possibilidades de resolução para a divisão



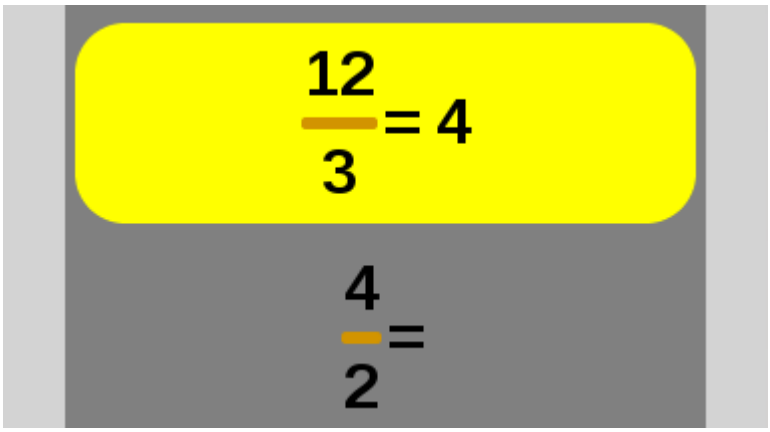
Fonte: Produzido pelo autor

Figura 15 – O jogador recebe 1 ponto pelo acerto da solução

Fonte: Produzido pelo autor

O fluxo seleção/resolução se repete.

Figura 16 – A nova expressão é apresentada ao usuário



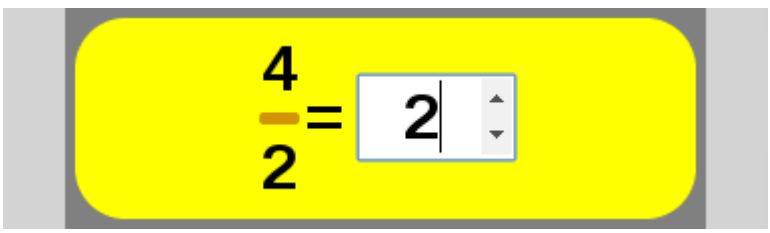
Fonte: Produzido pelo autor

A seleção da última operação não é pontuada.

Figura 17 – A seleção da última operação não é pontuada

Fonte: Produzido pelo autor

Figura 18 – O jogador insere a última solução a que está correta



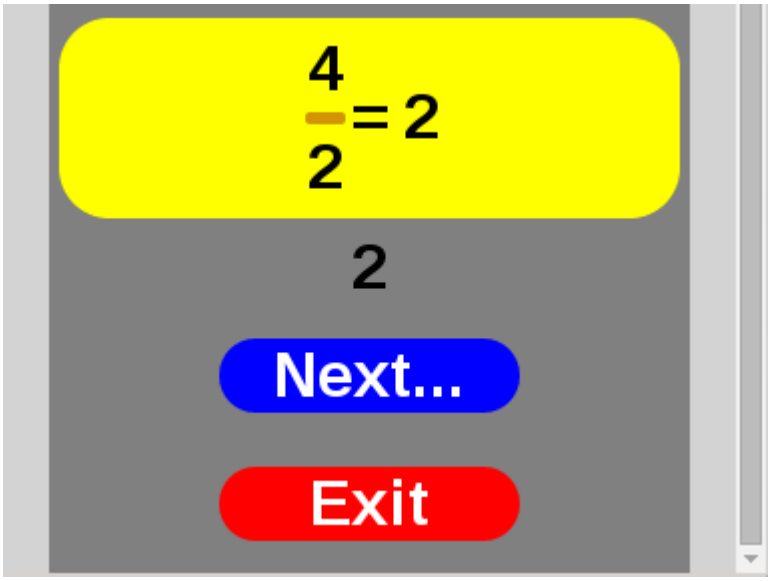
Fonte: Produzido pelo autor

Figura 19 – 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 20 – A tela final é apresentada



Fonte: Produzido pelo autor

9 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.

9.1 A linguagem formal envolvida no jogo

Vamos partir do conjunto de símbolos terminais que deve conter:

a) as operações de soma, subtração, multiplicação e divisão;

b) um conunto de números dado por um intervalo fechado sobre $\mathbb{Z}$;

c) 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.

9.1.1 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.

9.1.2 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:

a) quando uma subtração tem como operando direito outra operação que seja uma soma ou subtração;

b) quando o operando de uma multiplicação ou divisão for uma operação de soma ou subtração;

c) quando um número negativo é operando direito de uma operação.

9.1.3 Gramática referência

Tendo em conta os pontos apresentados anteriormente é possível criar uma gramática que seja capaz de gerar as expressões que são utilizadas no jogo. Algumas simplificações serão feitas a seguir nas produções: os terminais o e i representam os seguintes conjuntos de terminais respecitvamente as operações envolvidas e um intervalo fechado sobre $\mathbb{Z}$.

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[1] tal gramática é Livre de Contexto.

9.2 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

[1] Apresentada no tópico 1.1 Classificação de Chomsky

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.

9.2.1 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.

9.2.2 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.

9.2.3 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.

10 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.

10.1 Ambiguidade

A possibilidade de mais de uma solução para uma mesma expressão pode ser explicada por meio da propriedade associativa das operações[1]. Por exemplo, $3 + 4 + 5$ é uma expressão ambigua pois poderia ser resolvida em duas ordens:

$3 + (4 + 5) =$

$3 + 9 =$

12

, ou,

$(3 + 4) + 5 =$

$7 + 5 =$

12

As duas expressões iniciais são equivalentes a expressão apresentada $3 + (4 + 5) = (3 + 4) + 5 = 3 + 4 + 5$. A ausência de parênteses na última expressão mostra que não há precedência obrigatória entre as duas operações de soma, ou seja, que ela pode ser resolvida em diferentes ordens. Já a expressão $5 - (4 + 3)$ não é ambigua em relação à ordem de resolução e possui apenas uma forma de ser resolvida. Isto acontece pois a propriedade associativa não é valida para as operações envolvidas $(+, -)$. Os parênteses expressam uma ordem de precedência existente e desta forma são obrigatórios, se retirados o valor da expressão será alterado.

10.1.1 Rotação das árvores

Para as expressões envolvidas no jogo a ordem dos símbolos não pode ser alterada, o jogo não lida com a propriedade comutativa na análise de soluções. A ordem dos símbolos das expressões é dada pela ordem transversal dos nós[2]. Os parênteses não são nós da

[1] Explicada em 2. Operações no Conjunto dos Números Inteiros

[2] Explicada em 3. Árvores binárias ordenadas e rotações

árvore que representa a expressão e estão implicitos na estrutura da seguinte forma, se uma operação de multiplicação tem como um de seus operandos o resultado de uma operação de soma, a operação de soma envolvida deve estar dentro de parênteses para expressar a precedência existente entre as operações.

Como apresentado no capítulo 3, a operação de rotação sobre uma árvore binária não altera a ordem transversal dos nós. No caso de uma expressão sem precedência entre as operações podemos rotacionar a árvore, que representa a expressão, indeterminadamente alterando sua estrutura mas não o significado (operadores e operandos em sua ordem original). Porém, no caso de expressões com parênteses necessáriosa árvore não pode ser rotacionada livremente e as restrições em relação à estas rotações é dada pela lógica que explica a necessidade dos parênteses.

Observando os exemplos de expressões dados anteriormente ($3+4+5$ e $5-(4+3)$) podemos entender melhor as rotações e suas restrições. No caso da primeira expressão, devido a propriedade associativa da soma sobre o conjunto dos números inteiros, uma soma pode ser o operando da outra e vice-versa. Na expressão seguinte a soma é operando da subtração apenas (precedência expressa na árvore pela relação de pai e filho existente entre a subtração e a soma).

10.1.1.1 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.

10.1.1.2 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 (caso da segunda expressão utilizada como exemplo anteriormente, $5-(4+3)$).

10.1.2 Números de Catalan

Para um expressão que contenha apenas operações de multiplicação o número de ordens possíveis de solução é dado pelo n-ésimo número de Catalan, onde n é o número de operações.[3] Esta característica é consequência da propriedade associativa da multiplicação sobre o conunto dos números inteiros e pode ser extendida a uma expressão só com soma já que a soma também é associativa em $\mathbb{Z}$

Como as expressões a serem avaliadas envolvem operações de soma e subtração nem todas as rotações são possíveis, sendo assim para conhecer o número de soluções

[3] Característica explicada em 4. Números de Catalan

não é possível apenas utilizar o n-ésimo número de Catalan. Se particionarmos a árvore exatamente onde existem as restrições podemos conhecer o número de soluções para cara partição (subárvore) e então combiná-los para encontrar o número total de soluções para uma determinada expressão.

10.2 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 possíveis soluções.

Para facitilar a explicação do algoritmo um pseudocódigo e um mapa de execução do algoritmo estão disponíveis como ANEXO A e ANEXO B respectivamente. O pseudocódigo explica a lógica geral envolvida e não entra em detalhes mais específicos de funcionamento do algoritmo. O mapa de execução é um exemplo de uma instancia de execução do algoritmo para uma árvore com 3 nós internos.

O ponto de partida do algoritmo é uma expressão (lembrando que as expressões são representadas por árvores binárias), esta é a expressão avaliada pelo algoritmo. O primeiro laço que ocorre no algoritmo é a iteração de uma fila de soluções, que inicialmente possui apenas a expressão passada inicialmente ao algoritmo. No pseudocódigo este laço está na linha 5 - controlado na linha 7 quando a primeira solução saí da fila iterada -, a fila de soluções é representada por "Soluções a avaliar"e a expressão inicial por "Árvore". No mapa de soluções a fila pode ser vista como a primeira linha da tabela que contém as árvores que representam as soluções (S0, S1, S2, S3, S4), S0 é a solução inicial (ou expressão inicial) e a iteração ocorre da esquerda para a direita.

O laço seguinte , que está dentro do anterior, itera uma lista que contém as operações contidas na solução corrente (em relação ao laço no qual este está contido) S na ordem inversa da ordem que resulta da passagem de um algoritmo de busca em largura sobre S. Esta lista está representada no pseudocódigo por "Nós a avaliar"cuja iteração está na linha 12. No mapa de soluções estas iterações estão representadas nas colunas cinzas ("Nós") e a ordem da iteração ocorre de cima para baixo.

No começo do laço sobre as operações uma lista é criada a partir do produto cartesiano das soluções dos dois nós filhos desta operação (armazenadas em um mapa de soluções por nó). Esta lista é referida como "Pré-soluções"no pseudocódigo e como colunas azuis no mapa de execução. A avaliação do nó corrente N (da iteração sobre os nós) ocorre uma vez para cada elemento da lista de pré-soluções e este é o terceiro laço - encadeado no anterior. A iteração da lista de pré-soluções está na linha 16 do pseudocódigo e a ordem das iterações no mapa de soluções é de cima para baixo.

Para cada pré-solução iterada a árvore inicial é alterada de acordo e então N é testado para possíveis rotações horárias e anti-horárias, primeiro no sentido horário e depois no sentido anti-horário. Estas sequências de testes são os dois próximos laços (um para cada sentido de rotação), laços que estão na mesma linha de execução (dentro da iteração das pré-soluções). No pseudocódigo estes laços podem ser observados nas linhas 24 e 36. Já no mapa de execução cada rotação está representada pelas células amarelas e vermelhas (possíveis rotações horárias e anti-horárias respectivamente).

Caso N seja raíz cada possível rotação, horária ou anti-horária, será testata como uma possível nova solução. O teste de novas soluções é feito por meio da comparação de uma chave que representa cada possível árvore como uma determinada ordem dos nós. No pseudocódigo o mapa "Soluções"contém o conjunto de chaves para as soluções já encontradas e os testes para novas soluções estão nas linhas 28 e 40. Cada possível rotação contém a subárvore resultante no mapa de execução (e.g. a primeira rotação possível - anti-horária - ocorre no nó b da solução S0 e seu resultado é dado pela subárvore da solução S4 que começa em c).

10.2.1 Análise de complexidade e desempenho

Para a análise de complexidade do algoritmo vamos partir das instruções que são executadas menos vezes e chegar até as instruções que são executadas o maior número de vezes. As instruções que estão fora dos laços são executadas uma vez apenas e a constante g representa a quantidade destas instruções. C representa o número total de instruções executadas para uma árvore com n operações:

$$C = g + ...$$

Como esclarecido no tópico anterior a estrutura do algoritmo contém cinco laços principais. Seja n o número de operações envolvidas na expressão ($n \in \mathbb{Z}*$) o primeiro laço é executado s vezes, onde $s \in \mathbb{Z}$ e $1 \leqslant s \leqslant C(n)$ ($C(n)$ representa o n-ésimo número de Catalan). Sendo h o número de instruções executadas dentro deste primeiro laço apenas hc representa o total de instruções executadas dentro do laço em todas as suas iterações.

$$C = g + hc + ...$$

O laço imediamente seguinte é a iteração sobre as operações envolvidas na expressão e é executado n vezes para cada iteração do laço anterior, quantidade que pode ser expressa como cn. Seja i uma constante que representa o número de instruções que estão imediatamente abaixo do laço sobre as operações a quantidade dessas instruções executadas durante uma availação é dada por icn.

$$C = g + hc + icn + ...$$

Em seguida as pré-soluções são analisadas. O número de pré-soluções p varia de

acordo com a subárvore testada sendo que $1 \leqslant p \leqslant C(n-1)$. A constante j representa as instruções diretamente ligadas ao laço que itera as pré-soluções e $jcnp$ o quantidade total de execução destas instruções.

$$C = g + hc + icn + jcnp + ...$$

Os dois laços seguintes, testes de rotações, tem como o número total de iterações (somadas) r, onde $r \in \mathbb{Z}$ e $0 \leqslant r \leqslant n-1$. Como os dois laços têm o mesmo número de operações, k, a quantidade de instruções executadas nestes laços durante todo o algoritmo é dada por $kcnpr$. Sendo assim temos como consumo total:

$$C = g + hc + icn + jcnp + kcnpr$$

O algoritmo tem como característica então o consumo de tempo polinomial expresso pela fórmula apresentada anteriormente. Para testar o desempenho do algoritmo de avaliação de soluções 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 em anexo (ANEXO C e ANEXO D) 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.

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 (ANEXO C) mostra o comportamento polinomial da expressão. No gráfico cuja a escala é logarítmica (ANEXO D) podemos observar com maior clareza como os 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.

Parte IV

Considerações Finais

Este trabalho é um Trabalho de Graduação para o curso de Sistemas de Informação da Escola Superior de Engenharia e Gestão. 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. Tais análises podem mostrar erros mais comuns, quantidade de exercícios feitos, aproveitamento (acertos em relação a exercícios resolvidos). As 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.

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. Com este avanço não seria necessário a utilização de expressões predeterminadas, ou seja, não seria necessário um recurso humano para gerar tais expressões.

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 utilizando o serviço de repositórios GitHub em http://github.com/camiloperic/xpress.

Referências

ABRAED. *Um em cada 73 brasileiros estuda a distância*. Disponível em: <http://www.abraead.com.br/noticias.cod=x1.asp>. Acesso em: 25 jun. 2014. Citado na página 20.

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 25.

DAVIS, T. *Catalan Numbers*. 2006. Disponível em: <http://mathcircle.berkeley.edu/BMC6/pdf0607/catalan.pdf>. Acesso em: 11 ago. 2014. Citado na página 31.

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 25.

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 26.

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 36 e 37.

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 31.

W3C. *Introduction to HTML4*. Disponível em: <http://www.w3.org/TR/REC-html40/intro/intro.html>. Acesso em: 25 jun. 2014. Citado na página 37.

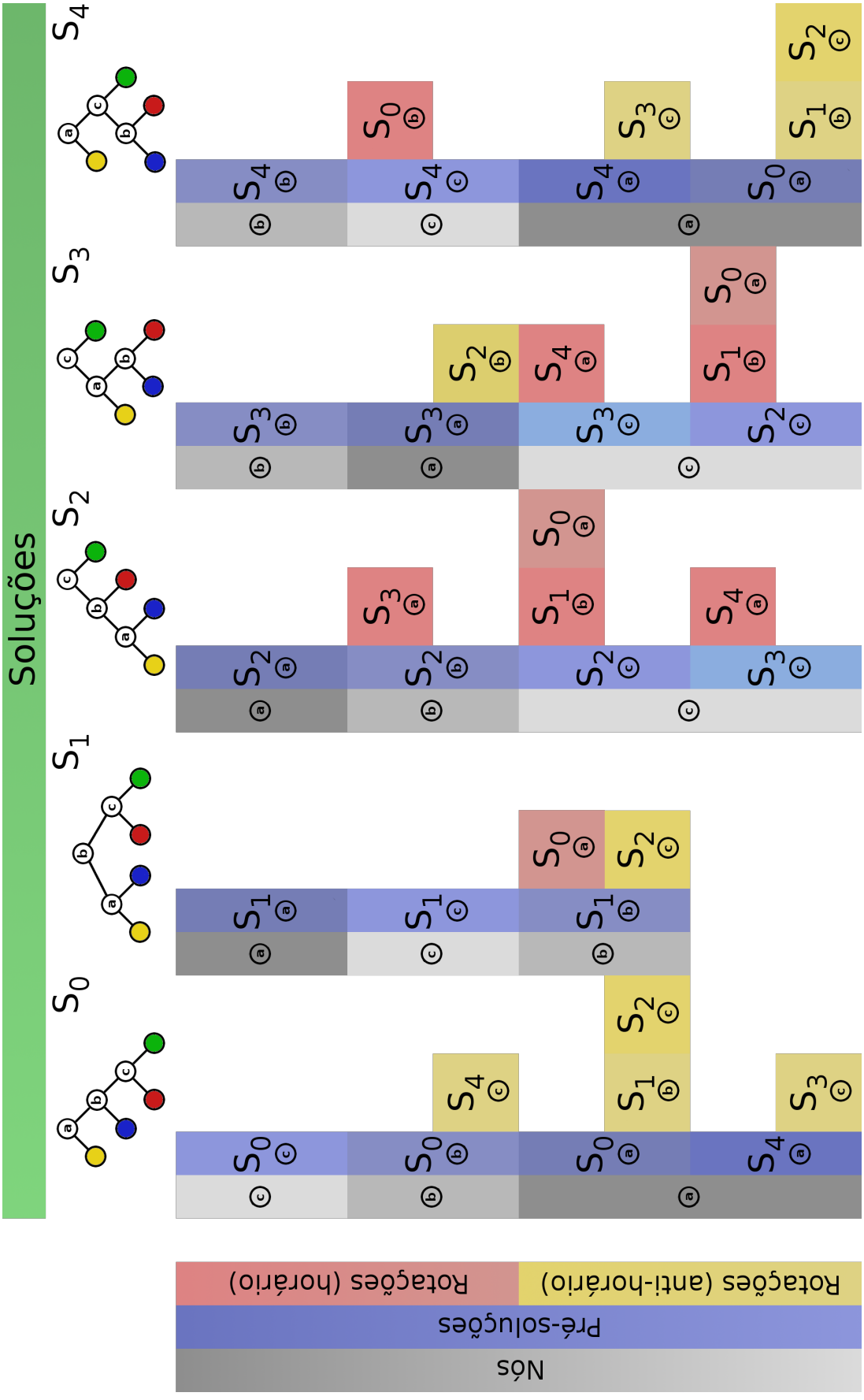
W3SCHOOLS. *HTML5 Introduction*. Disponível em: <http://www.w3schools.com/html/html5_intro.asp>. Acesso em: 25 jun. 2014. Citado na página 35.

Anexos

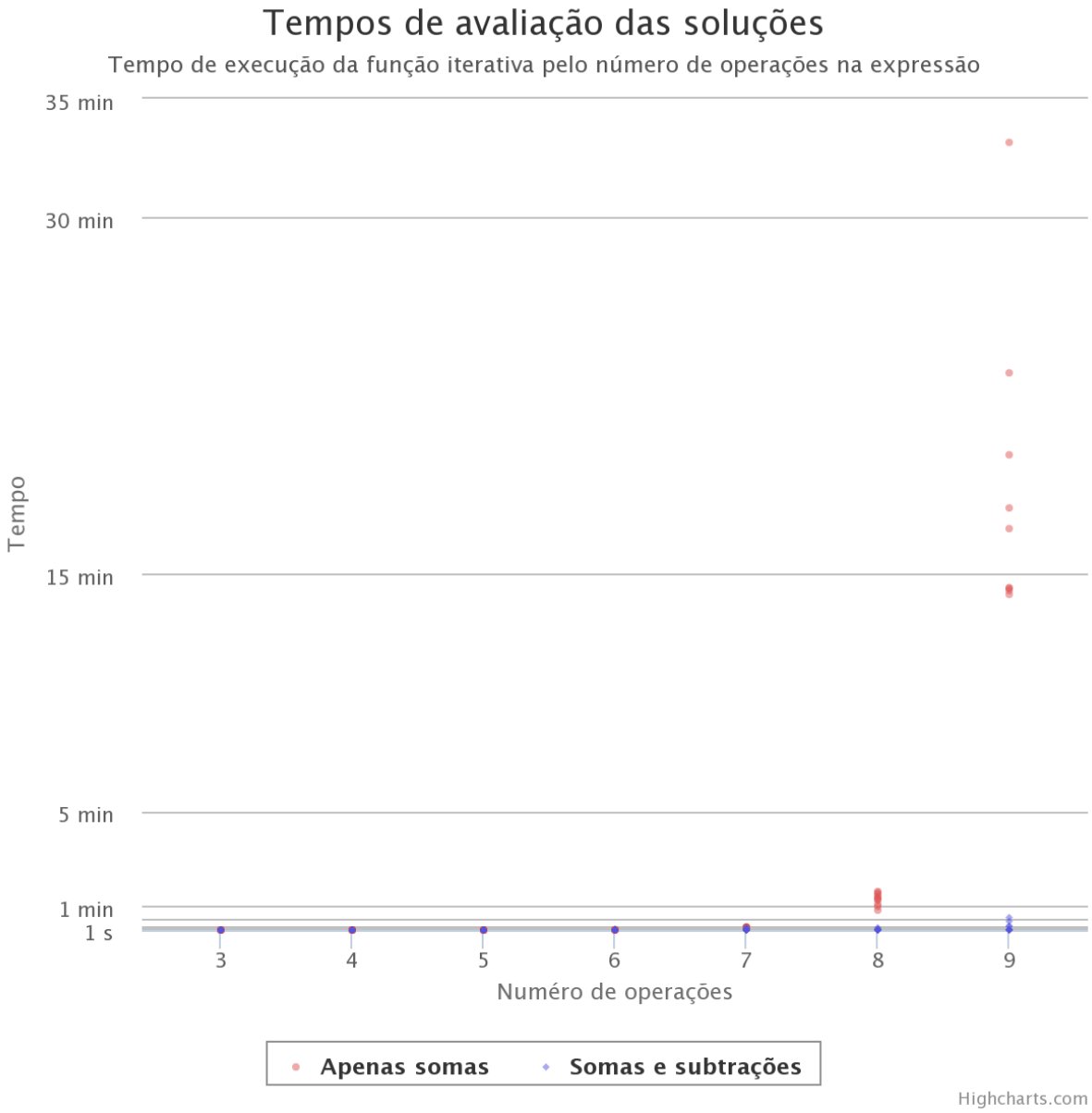
ANEXO A – Pseudocódigo

```
01  novo Mapa Soluções;
02  adicionar Árvore a Soluções;
03  nova Fila Soluções a avaliar;
04  enfilar Árvore em Soluções a avaliar;
05  enquanto Soluções a avaliar não é vazia {
06     nova Solução S1;
07     tirar a primeira Solução de Soluções a avaliar e atribuir à S1;
08     transformar Árvore utilizando as Transformações de S1;
09     nova Lista Nós a avaliar;
10     criar Lista de nós para a raíz de S1 e atribuir à Nós a avaliar;
11     novo Mapa Soluções por nó;
12     para cada Nó N em Nós a avaliar {
13        Soluções por nó em N recebe uma nova Lista;
14        nova Lista Pré-soluções;
15        combinar soluções dos filhos de N e atribuir à Pré-soluções;
16        para cada Solução P em Pré-soluções {
17           nova Solução S2 recebe a combinação de S1 e P;
18           adicionar S2 à Lista de N em Soluções por nó;
19           se N é raiz e a chave de S2 não está em Soluções {
20              adicionar S2 a Soluções e Soluções a avaliar;
21           }
22           novo Nó H recebe N;
23           nova Solução SH recebe S2;
24           enaquanto H pode ser rotacionado no sentido horário {
25              rotacionar H no sentido horário;
26              adicionar Transformação feita às Transformações de SH;
27              adicionar SH à Lista de N em Soluções por nó;
28              se o pai de H é raiz e a chave de SH não está em Soluções {
29                 adicionar SH a Soluções e Soluções a avaliar;
30              }
31              atribuir o pai de H à H;
32           }
33           desfazer todas as rotações feitas no sentido horário;
34           novo Nó AH recebe N;
35           nova Solução SAH recebe S2;
36           enaquanto AH pode ser rotacionado no sentido anti-horário {
37              rotacionar AH no sentido anti-horário;
38              adicionar Transformação feita às Transformações de SAH;
39              adicionar SAH à Lista de N em Soluções por nó;
40              se o pai de AH é raiz e a chave de SAH não está em Soluções {
41                 adicionar SAH a Soluções e Soluções a avaliar;
42              }
43              atribuir o pai de AH a AH;
44           }
45           desfazer todas as rotações feitas no sentido anti-horário;
46           desfazer as transformações de P;
47        }
48     }
49     desfazer as trasnformações de S1;
50  }
```

ANEXO B – Exemplo: mapa de execução



ANEXO C – Gráfico: Tempos de execução do algoritmo de avaliação de soluções (escala linear)



ANEXO D – Gráfico: Tempos de execução do algoritmo de avaliação de soluções (escala logarítmica)