UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

WEBIS:
Linguagem e ambiente Web acessível de programação

Lucas Pereira da Silva

Florianópolis
Julho de 2012

SUMÁRIO

1. INTRODUÇÃO
1.1. OBJETIVO GERAL
1.2. OBJETIVOS ESPECÍFICOS
2. DEFINIÇÃO DA LINGUAGEM BASEADA EM TELIS
2.1. GRAMÁTICA
2.2. GERAÇÃO DE CÓDIGO JAVASCRIPT
3. MÁQUINA WEBIS
3.1. EXCEÇÕES
REFERÊNCIAS BIBLIOGRÁFICAS

1. INTRODUÇÃO

A facilidade no aprendizado de uma primeira linguagem de programação está relacionada com a própria linguagem que é objeto de estudo e também com o ambiente de programação a ser utilizado. Boa parte das linguagens não possuem regras sintáticas e semânticas simples e tão pouco oferecem um ambiente de programação simplificado e intuitivo. Isso ocorre, pois na maioria dos casos, as linguagens de programação são desenvolvidas para serem ferramentas profissionais e não de aprendizado (GOMES; MENDES, 2012). Em decorrência desse fato, o primeiro contato com uma linguagem de programação pode ser muitas vezes traumático, fazendo com que o programador iniciante perca o interesse.

A linguagem Telis, desenvolvida no Laboratório de Software Educacional (Edugraf), possui regras sintáticas e semânticas simples que podem ser facilmente assimiladas. Telis também é o nome do ambiente de programação oferecido para a linguagem. Um programa Telis pode ser executado em um navegador Web através do uso de *applets* Java (PIERI *et al*, 2009).

SAVISKI (2010), propôs o porte da linguagem Telis para execução nativa em navegadores Web sem a necessidade da utilização de *applets*. A abordagem adotada foi a de gerar em JavaScript (linguagem interpretada nativamente pelos navegadores) o código escrito em Java da já existente máquina Telis.

O presente projeto terá embasamento no projeto desenvolvido por SAVISKI (2010) e consistirá do desenvolvimento de uma linguagem baseada em Telis e de uma máquina de execução escrita diretamente em JavaScript. O projeto também deverá ter grande foco no ambiente de programação que deverá ser executado nativamente por navegadores Web. O novo ambiente e a nova linguagem de programação a serem desenvolvidos serão chamados doravante de Webis.

A importância de um ambiente de programação acessível está no fato de possibilitar que qualquer pessoa possa aprender uma primeira linguagem de programação de maneira fácil e sem ter que se deparar com barreiras de acessibilidade. Garantir a acessibilidade é permitir que uma pessoa cega, por exemplo, possa utilizar o ambiente sem maiores dificuldades.

Projetos como o Eclipse Orion mostram que é possível realizar o desenvolvimento de um ambiente de programação que execute diretamente nos navegadores. O Eclipse Orion é um editor de texto para o desenvolvimento de aplicações Web escritas em HTML, JavaScript e CSS. Uma de suas principais ideias é fazer com que não só as aplicações executem na Web, mas também o

próprio ambiente de programação.

A utilização da Web como plataforma se encaixa muito bem nos requisitos de um ambiente acessível. Além de ser um sistema distribuído de baixo acoplamento, a Web se mostra uma ótima plataforma para a execução de aplicações, pois permite a manipulação de recursos *online* e gera independência da aplicação para com o sistema operacional. A Linguagem de Marcação de Hipertexto 5 (HTML5) introduz de maneira acessível novos recursos essenciais que ainda não eram suportados de forma padronizada pela Web. Isso garante ao usuário uma acessibilidade adequada e permite que ele se familiarize desde os primeiros passos no aprendizado de uma linguagem de programação com o desenvolvimento de aplicações que executem na Web.

1.1. OBJETIVO GERAL

Desenvolver uma linguagem de programação baseada em Telis e um ambiente acessível de programação. Tanto o ambiente de programação quanto os programas gerados pela linguagem deverão executar nativamente em navegadores Web sem a necessidade da utilização de extensões. A linguagem deverá priorizar a acessibilidade de programadores e usuários cegos possibilitando a utilização e manipulação de sons 3D que poderão ser usados para a criação de aplicações mais ricas e acessíveis.

1.2. OBJETIVOS ESPECÍFICOS

- Definir uma linguagem de programação com sintaxe e semântica baseada em Telis.
- Desenvolver o ambiente Web de execução para a linguagem definida.
- Desenvolver o ambiente Web de programação para a linguagem definida.
- Garantir a acessibilidade dos ambientes de execução e de programação desenvolvidos.
- Disponibilizar na linguagem elementos que permitam a utilização e manipulação de sons 3D.

2. DEFINIÇÃO DA LINGUAGEM BASEADA EM TELIS

A linguagem Telis que vem sendo desenvolvida desde 1988 (PIERI 2007) pelo Edugraf, permite, através de construções sintáticas e semânticas simples, que o programador tenha um grande poder de expressão e possa expor de forma natural conceitos avançados como orientação a objetos, paralelismo, programação distribuída, programação de eventos e outros.

A linguagem Webis será criada a partir da linguagem Telis. As construções léxicas e sintáticas da nova linguagem serão muito semelhantes as mesmas existentes em Telis e a diferença entre as duas linguagens estará principalmente relacionada com os novos requisitos da linguagem Webis. Por exemplo, como Webis deverá ter suporte a manipulação de sons 3D, será necessário que esses recursos sejam incorporados a nova linguagem.

Outra diferença entre a linguagem Telis e Webis estará na forma de comunicação utilizada. Em Telis é possível realizar a comunicação direta e imperativa da mesma forma que existe na orientação a objetos. É possível também realizar a comunicação através de estímulos baseados em tuplas. Webis usará apenas a segunda forma de comunicação.

A linguagem Webis, portanto, deverá ser constituída pela junção de um subconjunto de funcionalidades da linguagem Telis com um conjunto de novas funcionalidades específicas para a linguagem Webis. A sintaxe das duas linguagens, entretanto, será praticamente a mesma, tendo apenas algumas pequenas diferenças.

2.1. GRAMÁTICA

A linguagem Webis pode ser expressa através de uma gramática livre de contexto do tipo LL. Gramáticas do tipo LL possuem um conjunto adicional de restrições que permitem que a gramática possa ser analisada por um *parser* preditivo determinístico, ou seja, sem *backtracking*. Analisadores do tipo LL possuem ainda um parâmetro k chamado de *lookahed*. Esse parâmetro indica a quantidade de símbolos que serão levados em consideração em um passo do algoritmo. Quanto maior é o valor de k mais lento é o processo de análise e, em contra partida, menor é o número de não fatorações que uma produção da gramática pode conter. Uma gramática é dita fatorada se ela não possui produções

que derive sequências que inicie com o mesmo *token* por mais de um caminho.

Considerando um exemplo onde k é igual a um, então a gramática não pode ter nenhum símbolo não fatorado. Já se k é igual a dois, então a gramática pode conter apenas produções com no máximo um símbolo não fatorado.

A gramática do Webis descrita em EBNF é mostrada na Listagem 2.1.

```
1.  programa ::= (palavra | lista | COMENTARIO)*
2.  palavra ::=
3.      atribuicao |
4.      instanciacao |
5.      simbolo |
6.      texto |
7.      numero |
8.      operador |
9.      booleano
10. lista ::= "[" (palavra | lista)* "]"
11. simbolo ::= <IDENTIFICADOR>
12. atribuicao ::= <IDENTIFICADOR> "associar"
13. instanciacao ::= <IDENTIFICADOR> ("novo" | "nova")
14. texto ::= <TEXTO>
15. numero ::= <NUMERO>
16. operador ::= "+" | "-" | "=" | "*" | "/" | "\" | "^"
17. booleano ::= "verdadeiro" | "falso"
```

Listagem 2.1: Gramática EBNF do Webis.

Os itens entre parênteses angulares são definidos através de expressões regulares e foram omitidos para facilitar a leitura e diminuir o tamanho da gramática. Seus nomes são auto-explicativos e com exceção do `<IDENTIFICADOR>` dispensam maiores explicações. O `<IDENTIFICADOR>` é uma sequência de caracteres alfanuméricos que deve iniciar por uma letra e não contem espaços.

Em gramáticas do tipo LL(k), o valor mínimo de k é dependente da linguagem. Existem linguagens onde nem sempre é possível construir uma gramática que seja completamente fatorada, ou seja, onde k é igual a um. Esse é o caso do Webis que possui uma gramática do tipo LL(2). A não fatoração no Webis ocorre na produção `palavra`, pois nela é possível que um `<IDENTIFICADOR>` seja derivado por três caminhos diferentes: pelas produções `simbolo`, `atribuicao` e `instanciacao`.

Essa não fatoração gera um não determinismo, pois o analisador não sabe por qual caminho seguir quando encontra um `<IDENTIFICADOR>`. Para resolver esse problema é necessário que o algoritmo possa receber como entrada os dois próximos *tokens* e não apenas um. Com isso, ao perceber um `<IDENTIFICADOR>`, o analisador deverá ver qual é o próximo *token* para saber por qual produção a derivação irá seguir.

Apesar da gramática do Webis requerer um analisador do tipo LL(2) e este ser mais lento se comparado a um do tipo LL(1), isso não vem a ser um problema devido a boa performance dos computadores e analisadores atuais.

2.2. GERAÇÃO DE CÓDIGO JAVASCRIPT

As etapas de análise léxica, sintática e geração do código do Webis serão realizadas através da ferramenta ANTLR. O ANTLR é um gerador de analisador léxico e sintático que pode ser utilizado para a criação de interpretadores de linguagens, compiladores e outros tradutores (PARR 2007). No contexto desse projeto, o ANTLR será utilizado para gerar em JavaScript o código escrito em Webis. Nesse sentido, o ANTLR atuará como um tradutor e o código JavaScript será então executado pela máquina Webis.

Para realizar a geração de código JavaScript a partir de um código Webis é necessário entender alguns conceitos da arquitetura da máquina Webis que processará o código gerado. A máquina Webis será explicada em detalhes mais adiante, porém por hora é necessário compreender que cada `palavra` será traduzida para um elemento JavaScript e a máquina se encarregará de executar corretamente o elemento em questão conforme o seu tipo. Por exemplo, o `booleano` do Webis definido na Listagem 2.1 será traduzido para um elemento `Boolean` do JavaScript. O mesmo irá ocorrer com o `texto`, `numero` e `lista` do Webis que serão traduzidos para elementos JavaScript dos tipos `String`, `Number` e `Array`, respectivamente. Já o `simbolo`, a `atribuicao` e a `instanciacao`, ao contrário dos demais elementos, não serão traduzidos para tipos nativos do JavaScript, mas sim para os tipos próprios `Simbolo`, `Associar` e `Novo`, respectivamente. Por fim, um `operador` que semanticamente é tratado pela máquina Webis da mesma forma que um `simbolo`, também será traduzido para um objeto JavaScript do tipo `Simbolo`.

Ao analisar a gramática do Webis (Listagem 2.1), percebemos que um programa é composto por uma lista. Essa lista por sua vez é composta por palavras

e listas. A geração de código apenas irá traduzir isso para JavaScript. Com isso, a entrada da máquina Webis será uma lista JavaScript composta por números, textos, booleanos, listas e objetos.

3. MÁQUINA WEBIS

A máquina Webis trata-se basicamente de uma máquina de pilha onde os parâmetros necessários para as operações são desempilhados da pilha e o resultado das operações são empilhados na mesma. A entrada da máquina Webis nada mais é do que uma lista que contém os comandos do programa, sendo que cada um desses comandos é um elemento JavaScript que saberá se executar. O coração da máquina Webis, portanto, se torna muito simples. De forma simplificada, o que o núcleo da máquina Webis faz é iterar sobre a lista de comandos e enviar a ordem para cada um dos comandos se executar. Além da ordem de execução, o coração da máquina Webis fornece ao comando a ser executado o contexto do programa. Desta forma, cada comando tem acesso a pilha de dados e ao escopo de variáveis. Com isso, cada comando poderá empilhar e desempilhar elementos da pilha de dados e poderá também alterar, remover e criar variáveis.

A máquina Webis pode executar tanto dados quanto operações e símbolos. Quando um dado é executado ele simplesmente é colocado na pilha e fica disponível para ser consumido por uma operação ou símbolo. Quando operações ou símbolos são executados, eles podem ou não consumir elementos da pilha para produzir ou não algum resultado. Caso a operação produza algum resultado, então ele é colocado na pilha de forma que possa ser consumido por uma operação ou símbolo futuro.

No Webis, um símbolo pode ser uma primitiva que é uma operação básica fornecida pela linguagem, uma agenda de um ator ou uma variável. Quando a máquina encontra um símbolo, ela tenta executá-lo para algum desses casos. Caso a máquina não consiga executar o símbolo em questão, então uma exceção é lançada e em geral isso ocorre devido a erros de programação do usuário.

3.1. EXCEÇÕES

Devido ao fato da gramática Webis ser bastante simples, ocorrem poucas exceções em tempo de compilação. Em geral, a maioria das construções sintáticas são permitidas pela linguagem e os erros de compilação se limitam a: listas não fechadas, utilização de caracteres inválidos e textos não iniciados ou terminados por aspas. O código da Listagem 3.1 mostra um erro de compilação onde um texto não foi terminado por aspas.

```
"texto " "qualquer concatenar
```

Listagem 3.1: Exemplo de erro de compilação do Webis.

A maioria das exceções Webis ocorrem, no entanto, na etapa de execução e são gerados pela máquina Webis. As incoerências semânticas são tratadas todas durante o processo de execução e é por isso, que apenas exceções de erros sintáticos são lançadas durante o processo de compilação. O código da Listagem 3.2 mostra um erro de execução. O erro ocorre, pois na linguagem Webis o operador de exponenciação não pode ser aplicado a textos.

```
10 "texto" ^
```

Listagem 3.2: Exemplo de erro de excução do Webis.

REFERÊNCIAS BIBLIOGRÁFICAS

GOMES, Anabela; MENDES, A. J. **Learning to program**: difficulties and solutions. International Conference on Engineering Education, Coimbra, Portugal, 2007.

PIERI, Giovani *et al*. **Telis:** a programming tool set for beginners. *In*.: INTERNATIONAL INFORMATION AND TELECOMMUNICATION TECHNOLOGIES SYMPOSIUM, 8., 2009, Florianópolis, Santa Catarina, Brasil. Anais... Florianópolis: Fundação Bardal de Educação e Cultura, 2009. p. 183-186.

SAVISKI, Marcelo. **Porte de uma linguagem de programação para execução nativa em navegadores Web**. Curso de Ciências da Computação, Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis, 2010.

PIERI, Giovani. **Projeto e implementação de uma linguagem de programação**. Curso de Ciências da Computação, Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis, 2007.

PARR, Terence. **The definitive ANTLR reference**: Building domain-specific languages. The Pragmatic Bookshelf, 2007. 358 p.

CALDWELL, Ben *et al*. **Web Content Accessibility Guidelines 2.0**. World Wide Web Consortium, 2008. Disponível em: <http://www.w3.org/TR/WCAG/> Acesso em: dezembro de 2012.

XAVIER, Marcus Vinícius Cruz. **Telis ME**: uma versão de Telis para dispositivos móveis. Curso de Ciências da Computação, Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis, 2006.