Implementação de Ferramenta para Modelar Ambientes usando JavaCC

Rodolfo Berlezi

Orientador: Rogério Martins

Introdução

- JavaCompiler Compiler, é um gerador de analisadores para uso em aplicações Java
- Analisadores ou Parsers, são ferramentas que leem uma gramática específica
- E as converte para o programa Java que reconhece as ligações a gramática

Gramática

- São os elementos de uma língua e suas combinações.
- Um conjunto de regras e princípios que reagem ao uso de uma linguagem determinada.

Compiladores

- São os programas que traduzem o código fonte para um código objeto
- O processo é composto por analisadores

Analisador Léxico

- Separa no programa fonte cada símbolo que tenha algum significado para a linguagem
- Avisa quando um símbolo que não faz parte da linguagem é encontrado
- Ex: “O aluno will be escolhido”

  “Will be” não faz parte da gramática

Analisador Sintático

- Verifica se a sequência de símbolos existentes no programa fonte, forma um programa válido ou não
- É construído sobre uma gramática composta de uma série de regras que descrevem quais são as construções válidas da linguagem
- Ex: “Pública” válida

  “Plúbica” inválida

Analisador Semântico

- Verifica se não existem incoerências quanto ao significado das construções utilizadas
- Depende de uma tabela de símbolos
- Ex: “Ele é muito bom” coerente

  “Ele são muito bom” incoerente

Código

- A composição necessária para implementar um código em JavaCC é a seguinte:
- Declaração dos Tokens
- Declaração dos Skips
- Declaração dos Métodos

Tokens

```
TOKEN : /*OPERAÇÕES*/
{ < SOMA : "+">
| < SUBTRACAO : "-" >
| < DIVISAO : "/" >
| < MUTIPLICACAO : "*" >
}

TOKEN : /*OPERADORES*/
{ < DIGITO : ["0" - "9"] >
| < NUMERO : (< DIGITO >)+ >
}
```

Skips

```
SKIP :
{
    " "
|   "\r"
|   "\t"
|   "\n"
}
```

Métodos

```
double expr() :
{
  double a;
  double b;
}
{

  a = term()
  (
    "+" b = expr() {a = b;}
|   "-" b = expr() {a = b;}
  )*
  {return a;}

}
```

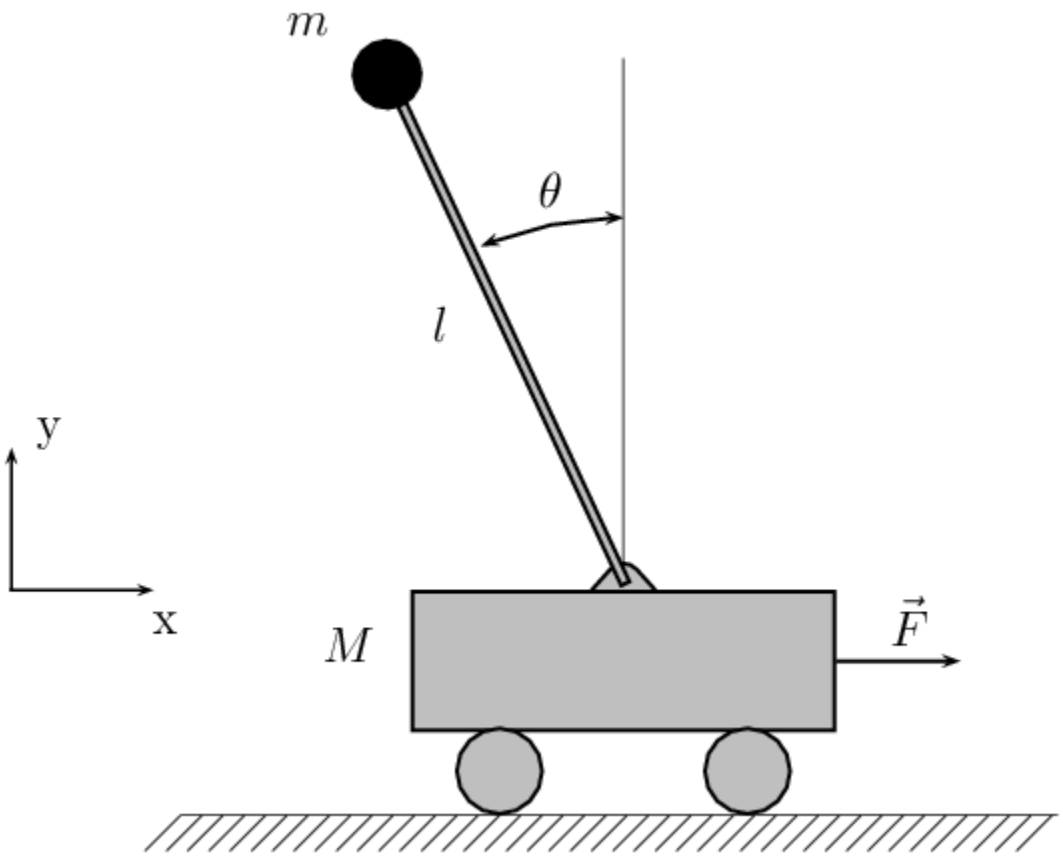
```
double term() :
{char op2;}
{
  unary()
  (
    (
      < MULTIPLY >  {op2='*';}
    | < DIVIDE >    {op2='/';}
    )

  {
    if(op2=='*') {
     resultado *= unary();
    }else{
       resultado /= unary();
    }
    return resultado;
  }

  )*

}
```

Pêndulo Invertido



Equação Diferencial

- Para representar o problema do Pêndulo Invertido, usaremos:
- Variáveis de posição para os ângulos do pêndulo
- Variável de posição para o carro
- Derivadas de velocidade de deslocamento para os ângulos e o carro

Interpretadores

- Para a aplicação será necessário o uso de três interpretadores, chamados de:
- Control
- Model
- View

Control

- Código que fará o controle
- Na forma de um Agente Evolutivo
- Através de Algoritmos Genéticos e Sistemas de Controles

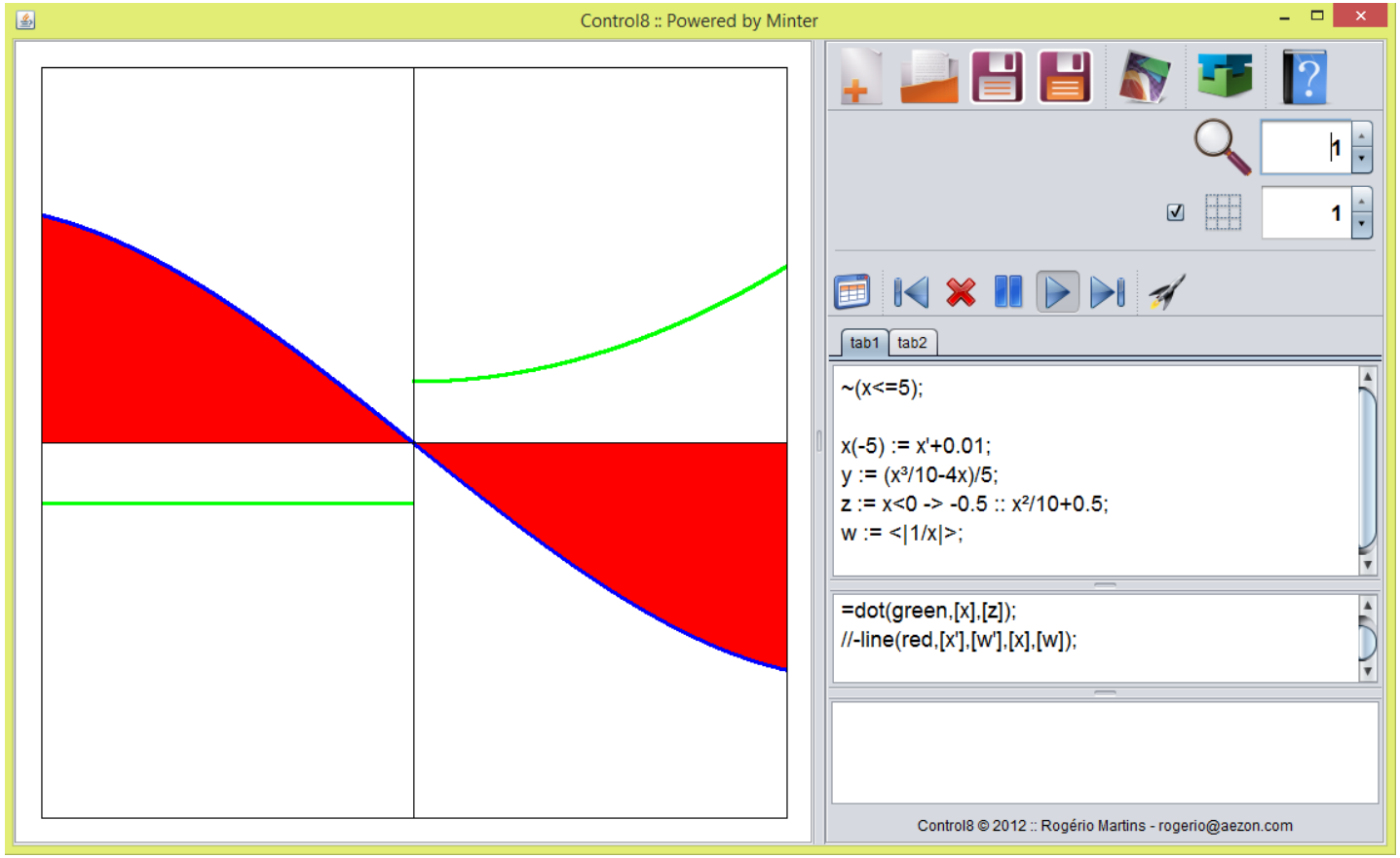
Model

- Ambiente controlado
- Receberá uma equação diferencial como parâmetro

View

- Poderemos observar o andamento do código e os seus resultados
- Sendo transpassado para a imagem gráfica posicionada ao lado

Aplicativo


Control8 :: Powered by Minter
1
1
tab1
tab2
~(x<=5);
x(-5) := x'+0.01;
y := (x³/10-4x)/5;
z := x<0 -> -0.5 :: x²/10+0.5;
w := <|1/x|>;
=dot(green,[x],[z]);
//-line(red,[x'],[w'],[x],[w]);
Control8 © 2012 :: Rogério Martins - rogerio@aezon.com

Obrigado por sua atenção!

Contato:

Rodolfo Berlezi

rodolfo_berlezi@hotmail.com

http://www.gca.unijui.edu.br/dodozaum

UNIJUÍ UNIVERSIDADE REGIONAL

Applied Computing Research Group