UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATISTÍCA
CURSO CIÊNCIAS DA COMPUÇÂO
PROFESSOR: Olinto José Varela

# ANALISADOR SINTÁTICO PREDITIVO LL(1)

**Lucas Pereira Da Silva (10100754)**

**Renan Oliveira Netto (10103126)**

Florianópolis
**Julho de 2012**

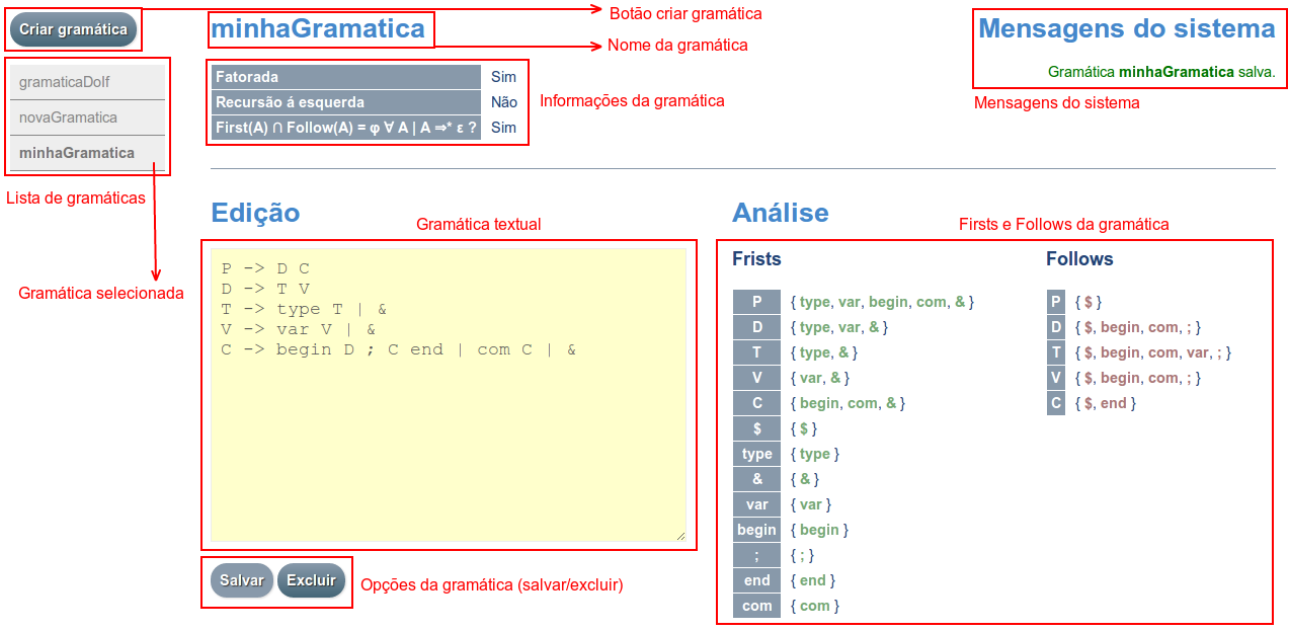
# Relatório

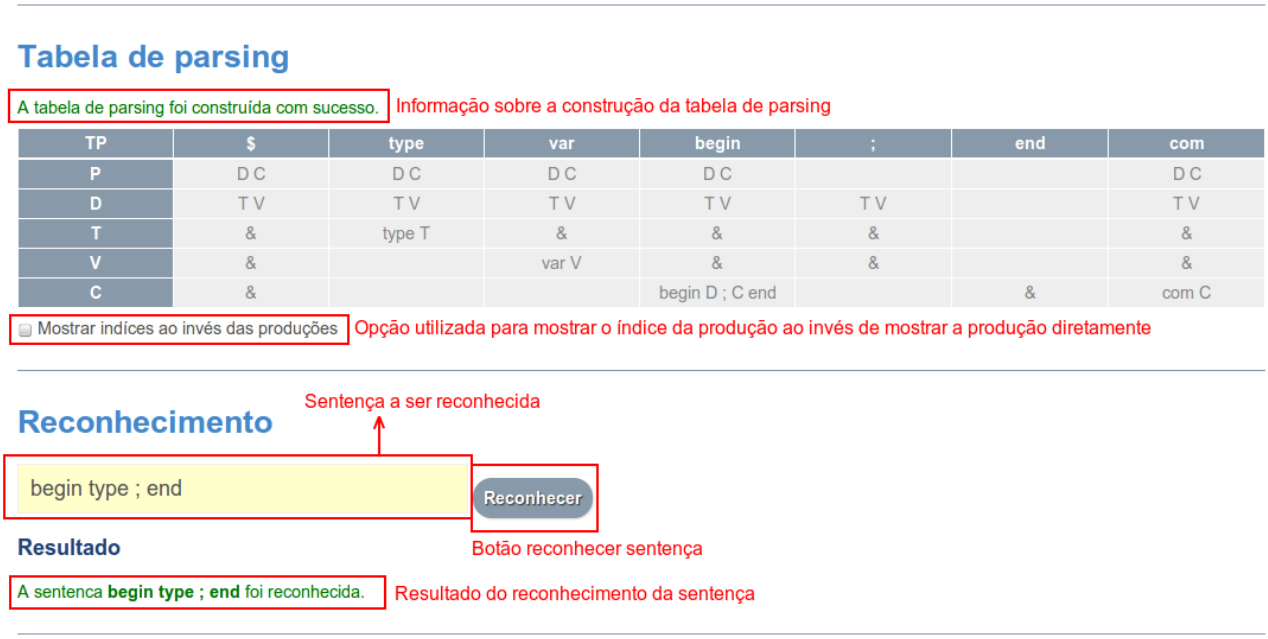
Escolhemos realizar a implementação do trabalho através da linguagem `JavaScript`. A escolha pela linguagem `JavaScript` se deu pelos mesmos motivos mencionados na trabalho anterior: o `JavaScript` é uma linguagem portável, de fácil manipulação e pode ser interpretada por qualquer navegador Web atual. Dessa forma, para a execução do nosso programa não é necessário compilar o código fonte e é possível executá-lo a partir de qualquer plataforma.

Para a execução do programa é necessário abrir o arquivo `livresDeContexto.html` que se encontra na pasta `aplicacao/html` que foi enviada junto com este trabalho.

O código fonte se encontra na pasta `aplicacao/js` e todos os arquivos-fontes se encontram documentados de forma que cada uma de suas funções estejam explicadas. Dividimos a implementação da nossa aplicação em módulos, sendo dois destes módulos responsáveis pela comunicação com a interface gráfica e pelo controle de interação com o usuário (arquivos `controleDoModelo.js` e `controleDaVisao.js`), um deles responsável por adicionar funcionalidades auxiliares para enriquecer o `JavaScript` (arquivo `utilitarios.js`) e por fim, o cérebro da nossa aplicação que é responsável por realizar todos os algorítimos, desde o cálculo de first e follow até a criação da tabela de parsing e reconhecimento de sentenças (arquivo `livresDeContexto.js`).

As imagens abaixo ilustram a aplicação desenvolvida e destacam as principais funcionalidades:






Um ponto interessante a mencionar na interface gráfica é a possibilidade de mostrar na tabela de parsing, tanto as produções quanto os índices das produções. Para alternar entre um modo e outro é necessário utilizar a caixa de marcação abaixo da tabela de parsing. Outro ponto importante é que a sentença a ser reconhecida deve ter os símbolos terminais separados por espaço. Se no caso da imagem acima fosse escrita a sentença **`begin type; end`** ao invés de **`begin type ; end`**, então a primeira sentença não seria reconhecida enquanto que a segunda seria.

O primeiro passo no desenvolvimento do trabalho foi realizar o cálculo dos firsts e follows. Decidimos realizar o cálculo dos firsts de forma recursiva. Cada símbolo não terminal chama os símbolos das suas produções e estes lhe fornecerão os seus firsts. Um ponto que causou problema nessa abordagem foi quando existia recursão a esquerda ou algum outro tipo de ciclo na gramática. Isso fazia com que a pilha de execução estourasse já que os não terminais iriam executar o método `calcularFirsts` indefinidamente. Para contornar esse problema, marcarmos os símbolos não terminais que já haviam sido chamados. Assim, ao passar novamente por esses símbolos (marcados), eles apenas retornavam os seus firsts que já haviam sido calculados.

O cálculo dos follows ocorreu de forma semelhante. A primeira parte do algorítimo foi determinar os follows que seriam derivados dos firsts. Essa parte foi feita de forma iterativa. Após isso, utilizamos um método para propagar os follows e esse método é executado pelos símbolos não terminais. Por exemplo, se `Follow(A)` pertence a `Follow(B)`, então o símbolo não terminal `A` irá propagar os seus follows para o não

terminal `B` e assim por diante. A recursão terminará quando um símbolo receber novos follows não houver nenhum follow novo dentre os recebidos.

Tendo o cálculo dos firsts e follows realizado, foi relativamente fácil realizar as outras operações. Para verificar se a gramatica é fatorada, utilizamos os firsts de cada produção. Caso a intersecção de firsts das produções de um símbolo não terminal não seja vazia, então a gramática não está fatorada.

Para verificar a recurção à esquerda, aproveitamos o cálculo do first para realizar essa tarefa. Se durante o cálculo do first um símbolo não terminal que ainda não finalizou o seu cálculo for chamado, então é porque ele é firsts de algum símbolo não terminal que por sua vez é first dele. Se isso acontecer, então o símbolo possui recursão à esquerda e consequentemente a gramática também.

Também não tivemos dificuldades para verificar se as intersecções dos firsts e follows dos não terminais que derivam **ε** são vazias.

Tendo todas as informações básica calculadas, partimos para a construção da tabela de parsing e posteriormente para o reconhecimento de sentenças. A construção da tabela de parsing foi fácil de ser implementada e seguiu o algorítimo visto em aula. Após a construção da tabela de parsing, o algorítimo de reconhecimento utilizou uma pilha e foi implementado sem maiores dificuldades.