Universidade Federal do Rio Grande do Sul
Departamento de Informática Aplicada
INF01147 – Compiladores
Prof. Lucas M. Schnorr - http://www.inf.ufrgs.br/~schnorr/

Especificação da Etapa 3 do Projeto de Compilador

Criação da Árvore Sintática Abstrata

A terceira etapa do trabalho de implementação de um compilador para a **Linguagem IKS** consiste na criação de uma árvore sintática abstrata (AST) baseada no programa de entrada, escrito em **IKS**. A árvore deve ser criada a medida que as regras semânticas são executadas e deve ser mantida em memória mesmo após o fim da análise sintática (ou seja, quando `yyparse` retornar, a árvore deve estar acessível). A avaliação deste trabalho será feita de duas formas: primeiro, através de uma análise visual da árvore, através da geração de um arquivo em formato `dot` definido pelo pacote GraphViz (funções serão fornecidas para tal através do repositório git do professor); segundo, por uma comparação automática da árvore gerada com aquela esperada para um determinado programa fonte.

1 Funcionalidades Necessárias

O resultado da terceira etapa deve ter as seguinte funcionalidades:

1. **Simplificar a especificação da gramática no caso das expressões aritméticas e lógicas**, de forma que as produções utilizem a forma mais simples que é inerentemente ambígua. Esta ambiguidade deve ser completamente tratada, considerando a precedência correta de operadores, através das ferramentas do bison para isso (veja a documentação sobre `%left`, `%right` ou `%nonassoc`).

2. **Remoção de conflitos *Reduce/Reduce*[1] e *Shift/Reduce*[2]** das regras gramáticas definidas pelo grupo na etapa anterior. Estes conflitos devem ser tratados através do uso das mesmas ferramentas bison do item precedente. Os mesmos podem ser observados através de uma análise cuidadosa do arquivo `parser.output` que será gerado automaticamente no momento da execução do make.

3. **Criar a árvore sintática abstrata** para uma entrada escrita em **IKS** qualquer, instrumentando a gramática com ações sintáticas ao lado das regras de produção descritas no arquivo `parser.y` para a criação dos nós da árvore e conexão entre eles. A árvore deve permanecer em memória após o fim da análise sintática. Neste item, o grupo deve reutilizar a implementação de árvore feita na etapa zero.

4. **Gerar o arquivo em format *dot* para análise gráfica e avaliação** utilizando as funções fornecidas pelo professor para que o grupo possa visualizar a árvore sintática abstrata gerada. Essas funções estão no repositório, nos arquivos `gv.c` e `gv.h`.

5. **Dois programas utilizando a sintaxe da linguagem IKS** devem ser implementados e disponibilizados juntamente com a solução desta etapa. O grupo tem a liberdade de escolher qualquer algoritmo para ser implementado.

2 Descrição da Árvore

A árvore sintática abstrata, ou AST (*Abstract Syntax Tree*), é uma árvore n-ária onde os nós intermediários representam símbolos não terminais, os nós folha representam tokens presentes no programa fonte, e a raiz representa o programa corretamente analisado. Essa árvore registra as derivações reconhecidas pelo analisador sintático, e torna mais fáceis as etapas posteriores de verificação e síntese, já que permite consultas em qualquer ordem. A árvore é abstrata porque não precisa representar detalhadamente todas as derivações. Tipicamente serão omitidas derivações intermediárias onde um símbolo não terminal gera somente um outro símbolo terminal, tokens que são palavras reservadas, e todos os símbolos "de sincronismo" ou identificação do código, os quais estão implícitos na estrutura reconhecida. Os nós da árvores serão de tipos relacionados aos símbolos não terminais, ou a nós que representam operações diferentes, no caso das expressões.

[1] `http://www.gnu.org/software/bison/manual/html_node/Reduce_002fReduce.html`
[2] `http://www.gnu.org/software/bison/manual/html_node/Shift_002fReduce.html`

2.1 Nó da AST

Cada nó da AST tem um tipo associado, e este deve ser um dos tipos declarados no arquivo `iks_ast.h` disponibilizado (veja seção 3). Quando o nó da AST é do tipo `IKS_AST_IDENTIFICADOR`, `IKS_AST_LITERAL`, ou ainda `IKS_AST_FUNCAO`, ele deve conter obrigatoriamente um ponteiro para a entrada correspondente na tabela de símbolos. Além disso, cada nó da AST deve ter uma estrutura que aponte para os seus filhos.

3 Controle e Organização da Solução

A função `main` deve estar em um arquivo chamado `main.c`. Outros arquivos fontes são encorajados de forma a manter a modularidade do código fonte. A entrada para o *bison* deve estar em um arquivo com o nome `parser.y`. A entrada para o *flex* deve estar em um arquivo com o nome `scanner.l`.

3.1 Git e Cmake

A solução desta etapa do projeto de compiladores deve ser feita sobre a etapa anterior. Cada ação de commit deve vir com mensagens significativas explicando a mudança feita. Todos os membros do grupo devem ter feito ações de commit, pelo fato deste trabalho ser colaborativo. Estas duas ações – mensagens de commit e quem fez o commit – serão obtidas pelo professor através do comando `git log` na raiz do repositório solução do grupo. Os arquivos adicionais necessários para esta etapa podem ser obtidos através do seguinte comando:

```
$ git pull origin master
```

O arquivo `parser.y` deverá ser novamente modificado para atender aos requisitos desta etapa. A solução do grupo deve partir deste código inicial, juntamente com o resultado da etapa anterior do projeto de compilador do mesmo grupo. Novos arquivos de código fonte podem ser adicionados, modificando o arquivo `CMakeLists.txt`, para que ele seja incluído no processo de compilação do analisador sintático.

3.2 Documentação do Código e Testes

Todas as funções devem estar documentadas. A escolha do sistema de documentação fica a critério do grupo e será igualmente avaliada. Uma opção é utilizar `doxygen`. A solução deve vir acompanhada com testes exaustivos que verificam o bom funcionamento da solução desta etapa.

4 Atualizações e Dicas

Verifique regularmente o Moodle da disciplina e o final deste documento para informar-se de alguma eventual atualização que se faça necessária ou dicas sobre estratégias que o ajudem a resolver problemas particulares. Em caso de dúvida, não hesite em consultar o professor.