**Implementação da análise semântica: a terceira etapa na construção do compilador Marvel**

**Cláudio Lopes (peso ), Gildo Leonel (peso ), Sérgio Rossini (peso )**

**Departamento de Ciência da Computação (DCC)**

**Universidade Federal de Juiz de Fora (UFJF)**

**Campus Universitário – CEP 36036-330 – Juiz de Fora – MG**

**{cslopes, gildo.leonel, sergiorossini}@gmail.com**

**1. Visão geral**

A fase de análise semântica de um compilador conecta as definições das variáveis com sua utilização, verifica se cada expressão possui um tipo correto e traduz a análise sintática para uma representação mais simples visando a geração de código de máquina [Appel e Palsberg, 2002].

**1.1. Desenvolvimento do trabalho**

**1.2. Módulos do analisador semântico**

O analisador semântico, como visto na seção 1, deve verificar as regras semânticas e iniciar a geração do código intermediário respectivo para cada nó da árvore de sintaxe abastrata. Dessa forma, os módulos AnalisadorSemantico e GeradorCodigo foram adicionados ao projeto do compilador.

O módulo AnalisadorSemantico é composto por classes que implementam a verificação das regras semânticas de verificação de escopo, verificação de tipos e verificação de declaração de variáveis. As classes foram nomeadas respectivamente como VerificadorEscopo, VerificadorTipos e VerificadorVariaveis.

O módulo GeradorCodigo é o início de um módulo maior. Nesse primeiro momento o módulo é responsável por gerar o código intermediário de cada nó pertencente à árvore de sintaxe abstrata criada durante a análise sintática.

Os módulos referentes à análise semântica e geração de código intermediário estão representados pela figura 1, onde também são mostradas as suas interdependências.

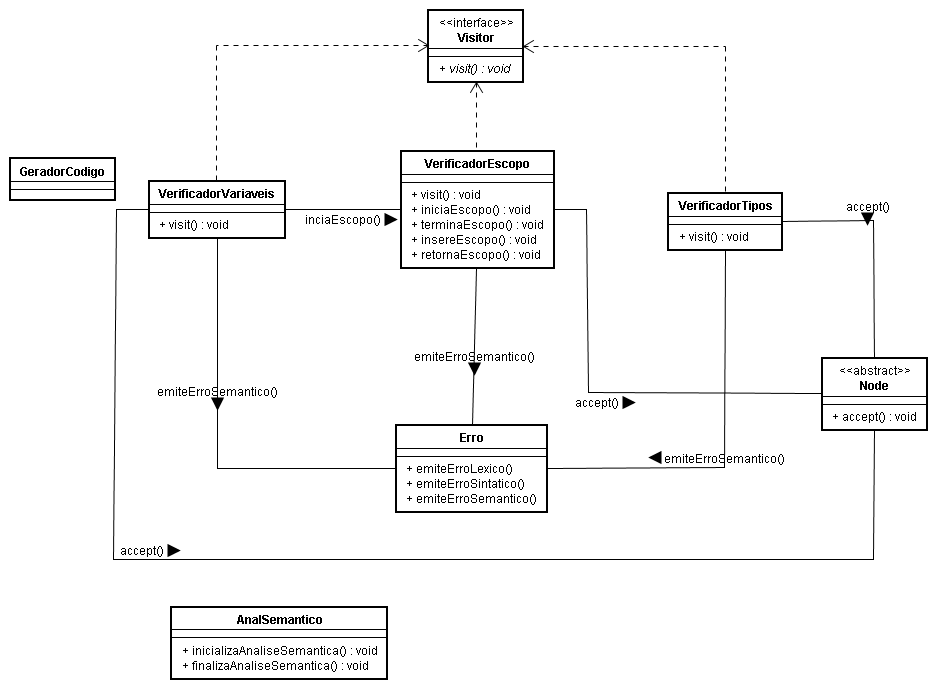


Figura 1. Interdependencia entre os módulos da análise semântica.

**2. Regras do analisador semântico**

As verificação das regras semânticas foram incluídas como forma de classes.

A verificação de tipos analise se seguintes regras foram incluídas para o compilador: verificação de tipos compatíveis entre variáveis durante

**3. Tabela de símbolos**

**4. Gerador de representação intermediária**

**5. Gerenciador de erros semântico**

**6. Resultados experimentais**

**6.1. Arquivo CODIGO.FRA**

**7. Conclusão**

**8. Referências bibliográficas**

Kernighan, B. e Ritchie, D. “The C Programming Language”, Prentice Hall Software Series, 2nd edition, capítulos 1, 2, 4, 5, 6 and 7.

Aho, A. e Ullman, J. (1995) Compiladores: Princípios, Técnicas e Ferramentas, Editora LTC, 1ª. Edição, capítulos 5, 6 e 7.

Menezes, P. (2000), Linguagens Formais e Autômatos, Editora Sagra Luzzato, 3ª. Edição, capítulo 3, subcapítulos 3.5 e 3.6.

Holzner, S. (2001). C++ Black Book, Editora Makron Books, 1ª. Edição, capítulos 5, 7 e 10.

Appel, A. W. e Palsberg, J (2002), Modern Compiler Implementation in Java, Cambridge University Press, 2nd. edition.

Deshpande, P. S. e Kakde, O.G., (2004), “C & Data Structures”, Charles River Media, 1st edition, capítulos 8, 10, 11, 12 e 15.

Vieira, M. (2007) “Projeto e implementação de um compilador – 3ª. parte”, Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas, Departamento de Ciência da Computação.

Vieira, M. (2007) “Teoria dos Compiladores – Notas de Aula”, Universidade Federal de Juiz de Fora, Instituto de Ciências Exatas, Departamento de Ciência da Computação.