CES-41: Compiladores

Laboratório 5: Geração do código intermediário para uma linguagem de programação, usando a ferramenta Yacc

Cássio dos Santos Sousa, Renan Pablo Rodrigues da Cruz

Professor: Fábio Carneiro Mokarzel

27 de dezembro de 2015

1. Introdução

Já foi possível criar, em laboratórios anteriores, um analisador léxico (com uso da ferramenta *Flex*), um analisador sintático, além do analisador semântico (com uso da ferramenta *Yacc*) para a Linguagem COMP-ITA 2015.

Para este laboratório, como o título alerta, o objetivo é o de gerarar o código intermediário para a mesma linguagem com o uso das mesmas ferramentas, de tal forma que o programa resultante seja capaz de imprimir o conteúdo da tabela de símbolos, as devidas mensagens de erros para programas quaisquer utilizados como entrada, além, é claro, de todas as quádruplas relativas ao código intermediário gerado.

Este foi um dos laboratórios mais complexos, de tal forma que seu desenvolvimento pode ser feito agora por duplas de alunos, contanto que essa dupla seja mantida até a entrega do último laboratório da disciplina.

2. Resultados

2.1. Atividades realizadas

A primeira das atividades foi a revisão do código escrito no laboratório anterior, o qual realizava a análise semântica e, caso não apresentasse erros, apresentava o código de entrada no formato *pretty printer*, removendo comentários, respeitando tabulações e espaçamentos de cada símbolo presente no código, além de, é claro, mostrar a tabela de símbolos. A revisão deu foco a uma impressão menos espaçada das informações do código, algo mais próximo do que é visto na linguagem C.

A segunda delas foi a adequação do código à Prática 4 (quando foi proposto o Laboratório 5) e à Aula 7 (que fazia o tratamento teórico do mesmo tema). Não foi possível apenas copiar e colar o código que estava presente, pois eles eram válidos para linguagens diferentes da linguagem COMP-ITA 2015. O foco foi o de justamente atentar às necessidades de cada trecho de código nas apresentações e quais adaptações eram necessárias para encaixá-los no nosso código.

A terceira atividade foi a verificação da impressão correta das quadruplas de código intermediário para um código que, a princípio, não possuísse erros sintáticos e/ou semânticos na linguagem COMP-ITA 2015. O código em questão foi justamente aquele dado como exemplo nas especificações da linguagem. A saída do código implementado para este laboratório imprime, primeiramente, o código inserido em conjunto com erros sintáticos e semânticos, e se não houver mais impedimentos, a tabela de símbolos e, em seguida, as quádruplas que compõem o código intermediário correspondente.

A quarta atividade deu foco à criação de testes e revisão concomitante do código. Cada teste inserido tentou demonstrar um cenário de geração de quádruplas de acordo com as especificações da linguagem COMP-ITA 2015. Estes testes foram separados em pastas internas à pasta **`Tests`**, com nome e numeração adequados para facilitar a busca. O input está presente no formato **`lab05test#`**, e o output correspondente está presente em **`lab05results#`** na mesma pasta, semelhante ao que foi feito para os exemplos na atividade anterior.

2.2. Formato da Tabela de Símbolos

Manteve-se o formato da Aula 7 e da Prática 4 com respeito a estrutura das quadruplas referentes à descriminação do código intermediário.

O trecho a seguir foi extraído do arquivo **`Tests/4_while/lab05results4`**, para ilustrar o formato do código intermediário.

```
4) EQ, (VAR, i), (INT, 2), (VAR, ##2)
5) JF, (VAR, ##2), (IDLE), (ROTULO, 8)
6) ATRIB, (INT, 3), (IDLE), (VAR, i)
7) JUMP, (IDLE), (IDLE), (ROTULO, 10)
```

2.3. Detalhes notados

Uma coisa que se percebeu foi que é possível que uma função e um procedimento terminem suas tarefas sem retornar nada, dado que um Statement vazio pode ser utilizado num ReturnStat. Isso pode não trazer problemas para a main e para procedimentos, que não retornam valores, mas isso traz graves problemas para funções, pois a checagem pedida como teste semântico foi só aquela na qual uma função termina seus statements com return. Por conta do tempo e da não-especificação deste caso teste, ele foi desconsiderado de novo.

2.4. Casos de teste

Na pasta **`Tests`** dentro da pasta de códigos, existem os seguintes testes, acompanhados de seus respectivos resultados. Tentou-se utilizar o código exemplo o máximo possível nos testes:

- **`1_atribuicao:`** código para verificar a correta geração de quádruplas referentes à atribuição de variáveis;
- **`2_if_sem_else:`** código para verificar a correta geração de quádruplas referentes ao if statement sem else;
- **`3_if_com_else:`** código para verificar a correta geração de quádruplas referentes ao if statement seguindo de else;
- **`4_while:`** código para verificar a correta geração de quádruplas referentes ao comando while;
- **`5_for:`** código para verificar a correta geração de quádruplas referentes ao comando for;
- **`6_read:`** código para verificar a correta geração de quádruplas referentes ao comando read;
- **`7_write:`** código para verificar a correta geração de quádruplas referentes ao comando write;

3. Conclusões

Foi possível avançar bastante na construção do compilador esperado para a linguagem COMP-ITA 2015. A geração do código intermediário foi uma tarefa bem complicada, e sua complexidade pôde ser notada no grande número de quádruplas geradas menos para códigos curtos.

Espera-se que este laboratório tenha servido de conclusão, por meio não só de seu código, mas também de sua complexidade dentro da temática de compiladores.