PCS2056 – Linguagens e Compiladores

Assunto: Recuperação de erros em autômatos finitos e em autômatos de pilha estruturados.

Objetivo: Completar o estudo da sintaxe de linguagens de programação com a inclusão de mecanismos de recuperação de erros de sintaxe. Estudar a recuperação absoluta de erros simples e a recuperação de erros múltiplos por meio da técnica de "panic mode".

Palavras-chave:

ponto de ocorrência de erro
ponto de detecção de erro
região de influência do erro
erro simples
erros múltiplos
correção automática
recuperação absoluta
panic mode

Enunciado:

1) Quais são todos os erros simples que podem ocorrer no primeiro estado de um autômato que deve aceitar apenas a cadeia "abc"?

2) Como fica o autômato do exercício anterior se ele for modificado para aceitar também todos os casos incorretos previstos?

3) Repita o procedimento para os outros dois estados do reconhecedor da cadeia "abc". Como deve ser tratada a recuperação de erros no estado final do autômato finito?

4) Descreva o método para todos os estados de um autômato finito.

5) Estenda o procedimento para as sub-máquinas de um autômato de pilha estruturado. Como tratar as chamadas e os retornos de sub-máquinas? Sugestão: Usar look-ahead nas transições em vazio, inclusive nos estados finais.

6) "Panic Mode" é um método radical para recuperação de erros genéricos, e consta da remoção da parte da cadeia de entrada compreendida entre o ponto de detecção do erro e um ponto em que seja encontrado um símbolo de sincronização, ou seja, um elemento da cadeia de entrada que esteja ou possa ser associado com facilidade a um estado bem determinado do autômato. Nesse ponto força-se tal estado e prossegue-se o reconhecimento.

PCS2056 – Linguagens e Compiladores

Assunto: otimização – conceitos gerais

Objetivo: Apresentar formas e atividades ligadas à otimização do código em compiladores.

Conceitos e palavras-chave:

otimização
blocos básicos
otimização local
otimização global
otimização independente de máquina
otimização dependente de máquina
otimização de expressões aritméticas
otimização de expressões booleanas
flattening (agrupamento de somas/produtos)
folding (propagação de constantes)
eliminação de sub-expressões repetidas
otimização de loops

Questões e exercícios:

1) Como podem ser exploradas as principais propriedades da álgebra – por exemplo, associatividade, comutatividade, etc. – na otimização do código gerado para expressões aritméticas?
2) É sempre possível aplicar as propriedades comutativa e associativa? O que ocorre quando um dos operandos não se apresenta com um valor fixo (por exemplo, no caso de chamada de função)?
3) O que pode acontecer ao se aplicar a distributividade sobre operandos cujos valores são de ordens de grandeza muito díspares? Como contornar essa situação?
4) Monte um algoritmo que otimize o tempo de execução de uma expressão booleana através da substituição do cálculo de operações como AND, OR, etc., por testes e desvios equivalentes.
5) Seqüências de operações aditivas ou multiplicativas ajudam a aplicar a associatividade. Como identificá-las? Como transformar a árvore visando à otimização do código nesse caso?
6) Proponha uma forma de otimização para loops, considerando: expansão múltipla, eliminação de invariantes, e redução da complexidade de operações.

PCS2056 – Linguagens e Compiladores

Assunto: 6a. parte do projeto – integração das rotinas semânticas.
Entregar no dia 02/12/2008

Objetivo: Finalizar a construção do compilador através da introdução das rotinas semânticas no analisador sintático.

Palavras-chave:

características sensíveis ao contexto
manipulação da dependência de contexto
ações semânticas
geração de código
rotinas semânticas
uso da tabela de símbolos
uso de variáveis locais
uso de parâmetros
uso de pilhas semânticas
pilha implícita
símbolos de ação
gramática de atributos

Atividades de projeto:

1) Nesta parte do projeto espera-se que sejam inseridas as rotinas semânticas do analisador semântico no analisador sintático baseado em autômatos de pilha estruturados já desenvolvido para a linguagem de programação, anteriormente definida para ser implementada neste projeto.

2) Identificar na gramática desenvolvida na segunda parte do projeto quais são e os pontos de inserção das rotinas semânticas.

3) Implementar e finalizador o analisador semântico.

4) Complementar o analisador sintático.

5) Usar o arquivo um texto de entrada contendo um programa escrito na linguagem de programação definida pela gramática da segunda parte do projeto como teste.