Heaven and REO (8) - Otimização de Código

Mateus Carvalho Gonçalves - 201810245

Otimizações são transformações no código (intermediário ou final) buscando aumentar sua eficiência, por exemplo, de tempo e armazenamento. Utiliza de um conjunto de heurísticas para identificar padrões conhecidos que representam oportunidades de melhoria na estrutura do código, mas nem sempre essas transformações resultam em melhorias de desempenho, necessariamente.

Existem dois tipos de otimizações: independentes ou dependentes de máquina. A primeira não tem uma máquina alvo, logo, são transformações genéricas que não utilizam do conjunto de instruções de máquina, para que possam ser utilizadas em ambientes diversos, e são realizadas sobre o código intermediário. Já a otimização dependente de máquina, como o próprio nome diz, tem uma máquina alvo e utiliza do conjunto de instruções dela para isso, realizando as transformações geralmente sobre o código final, produzindo um resultado extremamente específico.

A Otimização é um processo dividido em etapas. Primeiro é necessário uma análise do código buscando as oportunidades de melhoria e só depois são feitas as transformações, de fato. O subprocesso de análise, chamado de Análise de Fluxo, também é dividido em duas etapas: Análise de Fluxo de Controle e Análise de Fluxo de Dados.

Dessa forma, para a Otimização a primeira etapa consiste na Análise de Fluxo de Controle. Ela recebe como entrada um código intermediário, e faz o particionamento dele em blocos básicos de instruções. Esses blocos são conjuntos em que seus elementos serão executados obrigatoriamente de forma sequencial. Assim, são iniciados em instruções iniciais de um procedimento, uma instrução alvo de um comando de desvio, e instruções imediatamente seguintes a um comando de desvio, enquanto são finalizados em instruções imediatamente anteriores à instruções iniciais. Concluindo essa etapa, deve ser criado um Diagrama de Controle de Fluxo, que define os possíveis fluxos de código, e para isso são criados blocos de entrada e saída - para garantir que o programa/procedimento tenha apenas um ponto de início e fim - e são adicionadas as arestas que definem as possibilidades de fluxo entre blocos.

Depois disso, a fase de Análise de Fluxo de Dados percorre o diagrama analisando a definição e uso dos dados no programa. Essa etapa é importante para entender a manipulação dos dados local e globalmente, para então finalizar com as transformações da otimização de fato, que também podem ser feitas em âmbito local e global.

As otimizações independentes de máquina geralmente trazem melhorias em tamanho de código e tempo de execução. De forma geral e simplista, buscam reduzir instruções inacessíveis, ou que são executadas mais de uma vez sem necessidade, etc., e uma das principais fontes de alvo de otimização são estruturas de repetição. Por exemplo:

Retirada de comandos de um loop:

`for (i = 0; i < n; i++) {`
`  a = j + 5;`
`  f(a*i);`
`}`
	`a = j + 5;`
	`for (i = 0; i < n; i++) {`
	`  f(a*i);`
	`}`

Eliminação de instruções repetidas:

`y = a*x + b;`
`z = 666;`
`y = a*x + b;`
	`y = a*x + b;`
	`z = 666;`

Alguns outros exemplos de otimizações independentes de máquina são: eliminação de subexpressões comuns, eliminação de código morto, movimentação de código e eliminação de propagação de cópia.

Bibliografia

Slides de aula do Prof. Maurício Souza. Disponível no Campus Virtual.

Linguagens e Compiladores - Geração de código, noções de otimização. UNIVESP. Disponível em https://www.youtube.com/watch?v=oVi91pFqHlQ&list=PLxI8Can9yAHdmCAoLW5KWCG5ZrnfwrWXV&index=9&t=0s.