Análise Semântica e Representação Intermédia

Compiladores, Aula Nº 22

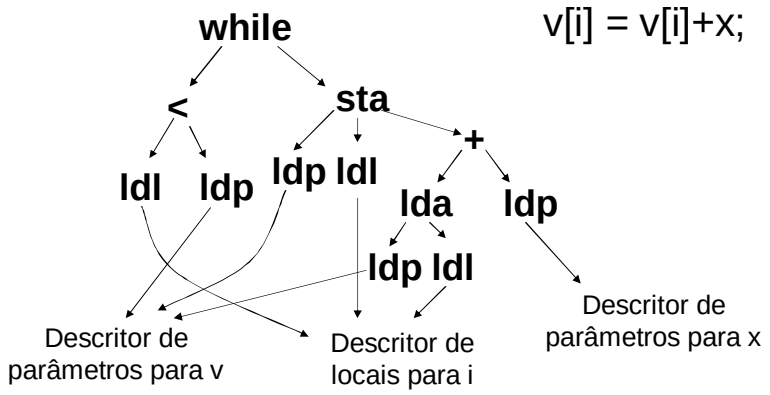
João M. P. Cardoso

Representação do fluxo de controlo

- Nós de enunciados
  - Nó **if**
    - Árvore de expressão para a condição
    - Nó para o corpo do then e nó para o corpo do else
  - Nó **while**
    - Árvore de expressão para a condição
    - Nó para o corpo
  - Nó **return**
    - Árvore de expressão para o valor/expressão de retorno

Exemplo

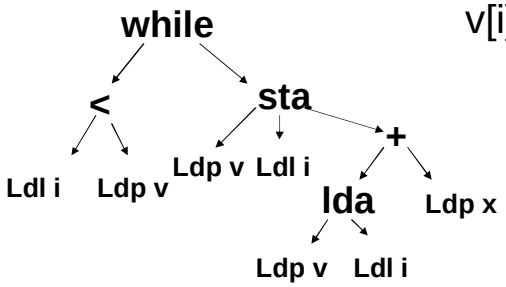
while (i < N)
v[i] = v[i]+x;



Exemplo

- Notação abreviada

while (i < N)
v[i] = v[i]+x;



Das árvores sintácticas até à IR

- Atravessar recursivamente a árvore sintáctica
- Construir representação de modo ascendente (*Bottom-Up*)
  - Ver identificador da variável na tabela de símbolos
  - Construir nós Load para aceder a variáveis
  - Construir expressões a partir dos nós de Load e dos nós de operações
  - Construir nós Store para enunciados de atribuição
  - Colocar nós while, if, return para as construções de controlo

Sumário

Representação Intermédia de alto nível

- Objectivo: representar o programa de um modo intuitivo para suporte de futuras tarefas de compilação
- Representação dos dados do programa
  - Tabelas de Símbolos
  - Organização hierárquica
- Representação da computação
  - Árvores de expressões
  - Vários tipos de nós store e load
  - Fluxo de controlo estruturado

Análise semântica: Erros

- Assumimos a inexistência de problemas durante a construção da IR
- Contudo, é necessário fazer muitas verificações durante a tradução
- Chamadas de **Análise Semântica**
- Realização da análise semântica ao nível da árvore sintáctica
  - Para que os erros sejam informativos/claros é necessário que os nós da árvore sejam anotados com as posições no programa

Objectivo da análise semântica

- Assegurar que o programa obedece a um conjunto de verificações de sanidade
  - Todas as variáveis usadas foram definidas
  - Tipos são usados correctamente
  - Chamadas a funções têm o número correcto e tipos de parâmetros e do valor retornado
- Verificação aquando da construção da IR

Descritores para identificadores

- Quando se constrói o descritor de uma variável local, parâmetro, etc. temos
  - Nome do tipo
  - Nome da variável
- O que é a verificação? Verificar se o nome do tipo identifica um tipo válido
  - *lookup* nome na tabela de símbolos dos tipos
  - Se não estiver lá, falha na análise semântica

Tabela de símbolos local

- Quando se constrói a tabela de símbolos local temos uma lista de descritores locais
- Fazer a verificação a quê?
  - Nomes de variáveis em duplicado
- Quando se faz a verificação?
  - Quando se insere o descritor na tabela de símbolos local
- Similar para os parâmetros, para a TS global, etc.

Verificação para loads, stores, etc.

- O que tem o compilador? Nome da variável.
- O que faz? *Lookup* nome da variável:
  - Se estiver na tabela de símbolos local, referencia descritor local
  - Se estiver na tabela de símbolos dos parâmetros, referencia descritor do parâmetro
  - Se estiver na tabela de símbolos global, referencia descritor global
  - Se não for encontrado um descritor: **erro semântico** (a variável não foi declarada...)

Verificação para a Instrução de Load para arrays

- O que tem o compilador?
  - Nome da variável
  - Expressão de indexação no array
- O que faz?
  - *Lookup* nome da variável (se não estiver: erro semântico)
  - Verifica tipo da expressão (se não for inteiro: erro semântico)

Operação de adição

- O que tem o compilador?
  - 2 expressões
- O que pode estar errado?
  - Expressões têm o tipo errado
  - Têm de ser os dois inteiros (por exemplo)
- Por isso o compilador faz a verificação do tipo das expressões
  - Instruções load guardam o tipo da variável acedida
  - Operações guardam o tipo da expressão produzida
  - Assim, é apenas necessário verificar tipos, e caso falhe dá erro semântico

Inferência de tipos para operações de adição

- Algumas linguagens deixam adicionar floats, ints, doubles
- Quais são os problemas?
  - Tipo do resultado da operação
  - Conversão dos operandos da operação
- Regras standard podem ser usualmente aplicadas
  - Se adição de um int e de um float
    - converter o int para float, adicionar os floats, e o resultado é um float.
  - Se adição de um float e de um double
    - converter o float para double, adicionar os doubles, e o resultado é um double.

Regras para a adição

- Princípio básico: Hierarquia de tipos de números (int, depois float, depois double)
- Todas as conversões forçadas são feitas de modo ascendente na hierarquia
  - Ex: int para float; float para double;
- Resultado tem o tipo do operando no nível mais elevado da hierarquia
  - int + float é float,
  - int + double é double,
  - float + double é double

Inferência de tipos

- Inferir tipos sem declaração explicita de tipos
- A adição é um caso muito restrito de inferência de tipos
- Tópico importante em investigação recente de linguagens de programação
  - Quantas declarações de tipos se podem omitir?
  - Ligado ao polimorfismo

Instrução Store

- O que tem o compilador?
  - Nome da variável
  - expressão
- O que faz?
  - *Lookup* nome da variável (se não estiver: erro semântico)
  - Verifica tipo da variável com o tipo da expressão
    - Se o tipo da variável não for compatível com o tipo da expressão, erro semântico

Instrução Store para arrays

- O que tem o compilador?
  - Nome da variável, expressão de indexação
  - expressão
- O que faz?
  - *Lookup* nome da variável (se não estiver: erro semântico)
  - Verifica se o tipo da expressão de indexação é inteiro
  - Verifica tipo da variável em relação ao tipo da expressão
    - Se o tipo da variável não for compatível com o tipo da expressão: erro semântico

Invocação de funções

- O que tem o compilador?
  - Nome da função, parâmetros
- Verificações:
  - Nome da função é identificado na tabela de funções do programa
  - Tipos dos parâmetros “casam” com tipos dos parâmetros da declaração da função

Sumário de verificações semânticas

- Realizar a verificação semântica durante a construção do IR
- Muitas verificações são para certificar que se constrói uma IR correcta
- Outras verificações correspondem a verificações simples de sanidade
- Cada linguagem de programação tem uma lista que deve ser verificada
- Pode reportar muitos erros potenciais durante a compilação