IMP - Uma Linguagem Imperativa Simples

Semântica Operacional

2006/2

Departamento de Informática Teórica - Instituto de Informática - UFRGS

Plano

Vamos estudar uma linguagem imperativa simples - IMP

- Sintaxe abstrata
- Semântica Operacional
- Sistema de Tipos (para um extensão de IMP)
- Semântica Denotacional
- Semântica Axiomática
- Relações entre várias semânticas

Começamos com Semântica Operacional

Entidades Sintáticas de IMP

- Num - literais inteiros
  - n
- Bool - literais boolenos
  - `true`, `false`
- Id - identificadores
  - $x, y, \ldots$
- Bexp - expressões booleanas
  - b
- Aexp - expressões aritméticas
  - a
- Com - comandos
  - C

Sintaxe Abstrata - Aexp

- Para expressões aritméticas - Aexp

$$\begin{array}{rcl} a & ::= & n \\ & | & x \\ & | & a_1 + a_2 \\ & | & a_1 - a_2 \\ & | & a_1 * a_2 \end{array}$$

- Variáveis não são declaradas
- Todas variáveis possuem um valor inteiro
- Não há efeitos colaterais

Sintaxe Abstrata - Bexp

Para expressões booleanas - Bexp

$$
\begin{array}{rcl}
b & ::= & \texttt{true} \\
& | & \texttt{false} \\
& | & a_1 = a_2 \\
& | & a_1 \leq a_2 \\
& | & \neg b \\
& | & b_1 \wedge b_2
\end{array}
$$

Sintaxe Abstrata - Com

Para comandos - Com

$$
\begin{array}{rcl}
C & ::= & \texttt{skip} \\
& | & x := a \\
& | & C_1 ; C_2 \\
& | & \texttt{if}\ b\ \texttt{then}\ C_1\ \texttt{else}\ C_2 \\
& | & \texttt{while}\ b\ \ C
\end{array}
$$

As regras de tipo estão embutidas na definição da sintaxe

assim como em expressões, parênteses podem ser necessários quando árvores de sintaxe abstrata são linearizadas

Semântica Operacional de IMP

- O significado de expressões de IMP depende dos valores das variáveis
- Semântica operacional abstrai certos aspectos da execução de um interpretador concreto
- O valor de variáveis em um dado momento (*estado* ou *memória*) é abstraido como sendo uma função de $\mathsf{Id} \rightarrow \mathbb{Z}$
- de agora em diante chamaremos o conjunto $\mathsf{Id} \rightarrow \mathbb{Z}$ de Σ
- Usamos $\sigma, \sigma', \ldots$ como metavariáveis sobre elementos de Σ

S.O. *Big-Step* de IMP

- $a, \sigma \Downarrow v, \quad b, \sigma \Downarrow v$ e $\quad C, \sigma \Downarrow \sigma'$
- A avaliação de expressões aritméticas e boolenas não produz efeitos colaterais
- A avaliação de comandos produz efeitos colaterais mas não produz um resultado direto
- o "resultado"de um comando é um novo estado: $C, \sigma \Downarrow \sigma'$
- A avaliação de um comando pode não terminar

Regras de Avaliação para Aexp

$$n, \sigma \Downarrow n$$

$$x, \sigma \Downarrow \sigma(x)$$

$$\frac{a_1, \sigma \Downarrow n_1 \qquad a_2, \sigma \Downarrow n_2}{a_1 + a_2, \sigma \Downarrow n_1 + n_2}$$

$$\frac{a_1, \sigma \Downarrow n_1 \qquad a_2, \sigma \Downarrow n_2}{a_1 - a_2, \sigma \Downarrow n_1 - n_2}$$

$$\frac{a_1, \sigma \Downarrow n_1 \qquad a_2, \sigma \Downarrow n_2}{a_1 * a_2, \sigma \Downarrow n_1 * n_2}$$

Regras de Avaliação para Bexp

$$\texttt{true}, \sigma \Downarrow \texttt{true}$$

$$\texttt{false}, \sigma \Downarrow \texttt{false}$$

$$\frac{a_1, \sigma \Downarrow n_1 \qquad a_2, \sigma \Downarrow n_2}{a_1 = a_2, \sigma \Downarrow n_1 = n_2}$$

$$\frac{a_1, \sigma \Downarrow n_1 \qquad a_2, \sigma \Downarrow n_2}{a_1 \leq a_2, \sigma \Downarrow n_1 \leq n_2}$$

As premissas das regras para $=$ e $\leq$ são julgamentos avaliados pelas regras para Aexp

Note que as regras acima não estabelecem ordem de avaliação

Regras de Avaliação para Bexp(cont.)

$$\frac{b_1, \sigma \Downarrow \texttt{false}}{b_1 \wedge b_2, \sigma \Downarrow \texttt{false}}$$

$$\frac{b_2, \sigma \Downarrow \texttt{false}}{b_1 \wedge b_2, \sigma \Downarrow \texttt{false}}$$

$$\frac{b_1, \sigma \Downarrow \texttt{true} \qquad b_2, \sigma \Downarrow \texttt{true}}{b_1 \wedge b_2, \sigma \Downarrow \texttt{true}}$$

- Regras acima não fixam ordem de avaliação
- Mas expressam o seguinte: na avaliação de uma conjunção, se um das expressões booleanas avaliar para `false` o resultado final será `false`.

Regras de Avaliação para Bexp(cont.)

$$\frac{b_1, \sigma \Downarrow \texttt{true}}{b_1 \vee b_2, \sigma \Downarrow \texttt{true}}$$

$$\frac{b_2, \sigma \Downarrow \texttt{true}}{b_1 \vee b_2, \sigma \Downarrow \texttt{true}}$$

$$\frac{b_1, \sigma \Downarrow \texttt{false} \qquad b_2, \sigma \Downarrow \texttt{false}}{b_1 \vee b_2, \sigma \Downarrow \texttt{false}}$$

- Regras acima também não fixam ordem de avaliação
- Expressam que na avaliação de uma disjunçao basta que uma das expressões avalie para `true` para que o resultado final seja `true`.

Regras de Avaliação para Com

$$\frac{}{\texttt{skip}, \sigma \Downarrow \sigma}$$

$$\frac{C_1, \sigma \Downarrow \sigma' \qquad C_2, \sigma' \Downarrow s''}{C_1; C_2, \sigma \Downarrow s''}$$

$$\frac{b, \sigma \Downarrow \texttt{true} \qquad C_1, \sigma \Downarrow \sigma'}{\texttt{if } b \texttt{ then } C_1 \texttt{ else } C_2, \sigma \Downarrow \sigma'}$$

$$\frac{b, \sigma \Downarrow \texttt{false} \qquad C_2, \sigma \Downarrow \sigma'}{\texttt{if } b \texttt{ then } C_1 \texttt{ else } C_2, \sigma \Downarrow \sigma'}$$

Regras de Avaliação para Com (cont.)

$$\frac{a, \sigma \Downarrow n}{x := a, \sigma \Downarrow \sigma[x \mapsto n]}$$

- $\sigma[x \mapsto n]$ é uma função definida da seguinte forma:

$$\begin{aligned} \sigma[x \mapsto n](x) &= n \\ \sigma[x \mapsto n](y) &= \sigma(y) \end{aligned}$$

Regras de Avaliação para Com (cont.)

$$\frac{b, \sigma \Downarrow \texttt{false}}{\texttt{while } b\ C, \sigma \Downarrow \sigma}$$

$$\frac{b, \sigma \Downarrow \texttt{true} \qquad C; \texttt{while } b\ C, \sigma \Downarrow \sigma'}{\texttt{while } b\ C, \sigma \Downarrow \sigma'}$$

Avaliação de Comandos - Observações

- A ordem de avaliação é importante e é especificada
  - C_1 é avaliado antes de C_2 em $C_1; C_2$
  - C_2 não é avaliado em `if true then` C_1 `else` C_2
  - C não é avaliado em `while false` C
  - b é avaliado primeiro em `if` b `then` C_1 `else` C_2
- Regras não são estruturais
  - Veja a regra para `while`
- Regras sugerem um interpretador
  - `if` e `while` possuem múltiplas regras mas somente uma pode ser aplicada em um dado momento

Desvantagem de S.O. *Big-Step*

Não nos dá uma forma de falar sobre estados intermediários

- Logo não pode ser usada para semântica de concorrência onde comandos podem ser executados de forma intercalada

Semântica *small-step* não possui essas limitações.
Execução é modelada como uma seqüencia (possivelmente infinita)

Semântica Operacional *Small-Step de IMP*

Para Com: vamos definir uma relação $C, \sigma \rightarrow C', \sigma'$

- C' é obtido a partir de C por um passo de execução atômico
- Avaliação termina quando o comando é reescrito para o comando "terminal" `skip`
- Alguns comandos nunca reduzem para `skip`: `while true` C

Granularidade do passo é escolha de quem define a semântica

S.O. *Small-Step* de Com

$$\frac{a, \sigma \rightarrow a', \sigma}{x := a, \sigma \rightarrow x := a', \sigma}$$

$$x := n, \sigma \rightarrow \texttt{skip}, \sigma[x \mapsto n]$$

$$\frac{C_1, \sigma \rightarrow C_1', \sigma'}{C_1; C_2, \sigma \rightarrow C_1'; C_2, \sigma'}$$

$$\texttt{skip}; C, \sigma \rightarrow C, \sigma$$

S.O. *Small-Step* de Com- (cont.)

$$\frac{b, \sigma \rightarrow b', \sigma}{\texttt{if } b \texttt{ then } C_1 \texttt{ else } C_2, \sigma \rightarrow \texttt{if } b' \texttt{ then } C_1 \texttt{ else } C_2, \sigma}$$

$$\texttt{if true then } C_1 \texttt{ else } C_2, \sigma \rightarrow C_1, \sigma$$

$$\texttt{if false then } C_1 \texttt{ else } C_2, \sigma \rightarrow C_2, \sigma$$

$$\texttt{while } b\ C, \sigma \rightarrow \texttt{if } b \texttt{ then } (C; \texttt{while } b\ C) \texttt{ else skip}, \sigma$$

Exercícios

1. Definir a S.O. *small-step* de Aexp e Bexp.
2. Dê outras estratégias para S.O. *big-step* de Bexp
3. Defina uma S.O. para o comando concorrente $C_1 \| C_2$.
   Justifique o estilo de S.O. adotado.
4. Supor que:
   - $C \equiv$ `while` $x > 1\ C_1$,
   - $C_1 \equiv\ r := r * x; x := x + 1$, e
   - σ é tal que $\sigma(x) = 2$ e $\sigma(r) = 60$

   verificar se $C, \sigma \rightarrow$ `skip`$, \sigma[r \mapsto 120][x \mapsto 1]$ (trabalhoso)