

Semântica Operacional Estrutural

52

Relação de Transição \Rightarrow

[ass_{sos}]

$\langle x := a, s \rangle \Rightarrow s[x \mapsto \mathcal{A}[\![a]\!]s]$

[skip_{sos}]

$\langle \text{skip}, s \rangle \Rightarrow s$

54

Semântica Operacional Estrutural

- O ênfase desta semântica é nos *passos individuais* de execução de um programa.
- A relação de transição tem a forma $\langle S, s \rangle \Rightarrow \gamma$ e representa o *primeiro passo* de execução do programa S no estado s.
- Em $\langle S, s \rangle \Rightarrow \gamma$, se γ é da forma $\langle S', s' \rangle$ então a execução ainda não está completa (é uma configuração intermédia).
 s' então a execução terminou e o estado final é s' .
- Uma configuração $\langle S, s \rangle$ diz-se *bloqueada* se não existir nenhuma configuração γ tal que $\langle S, s \rangle \Rightarrow \gamma$

53

Relação de Transição \Rightarrow

$$\begin{array}{c} [\text{comp}_{\text{sos}}^1] \quad \frac{\langle S_1, s \rangle \Rightarrow \langle S'_1, s' \rangle}{\langle S_1; S_2, s \rangle \Rightarrow \langle S'_1; S_2, s' \rangle} \\ \\ [\text{comp}_{\text{sos}}^2] \quad \frac{\langle S_1, s \rangle \Rightarrow s'}{\langle S_1; S_2, s \rangle \Rightarrow \langle S_2, s' \rangle} \end{array}$$

55

Relação de Transição \Rightarrow

[if_{sos}^{tt}]

$\langle \text{if } b \text{ then } S_1 \text{ else } S_2, s \rangle \Rightarrow \langle S_1, s \rangle \text{ if } \mathcal{B}[b]s = \text{tt}$

[if_{sos}^{ff}]

$\langle \text{if } b \text{ then } S_1 \text{ else } S_2, s \rangle \Rightarrow \langle S_2, s \rangle \text{ if } \mathcal{B}[b]s = \text{ff}$

56

Relação de Transição \Rightarrow

[while_{sos}]

$\langle \text{while } b \text{ do } S, s \rangle \Rightarrow$

$\langle \text{if } b \text{ then } (S; \text{while } b \text{ do } S) \text{ else skip}, s \rangle$

57

Sequência de Derivação

- Uma **sequência de derivação** de um programa S num estado s é uma de duas coisas:
 - uma sequência **finita** de configurações $\gamma_0, \gamma_1, \gamma_2, \dots, \gamma_k$ tais que $\gamma_0 = \langle S, s \rangle$, $\gamma_i \Rightarrow \gamma_{i+1}$ com $0 \leq i < k$, $k \geq 0$, e. γ_k é uma configuração terminal ou bloqueada;
 - uma sequência **infinita** $\gamma_0, \gamma_1, \gamma_2, \dots$ tais que $\gamma_0 = \langle S, s \rangle$ e $\gamma_i \Rightarrow \gamma_{i+1}$ para $0 \leq i$.
- $\gamma_0 \Rightarrow^i \gamma_i$ indica i passos de execução.
- $\gamma_0 \Rightarrow^* \gamma_i$ indica um número finito de passos de execução.

58

Sequência de Derivação

Exemplo: Seja s_0 o estado que mapeia todas as variáveis em 0, excepto x e y . Nestes casos: $s_0[x = 5]$ e $s_0[y = 7]$.

Para o programa $(z := x; x := y); y := z$

temos a sequência de derivação

$$\begin{aligned} & \langle (z := x; x := y); y := z, s_0 \rangle \\ & \Rightarrow \langle x := y; y := z, s_0[z \mapsto 5] \rangle \\ & \Rightarrow \langle y := z, (s_0[z \mapsto 5])[x \mapsto 7] \rangle \\ & \Rightarrow ((s_0[z \mapsto 5])[x \mapsto 7])[y \mapsto 5] \end{aligned}$$

Note que para **cada passo** da sequência temos uma **árvore de derivação** que justifica esse passo.

59

Árvore de Derivação

Por exemplo, o primeiro passo

$$\langle (z := x; x := y); y := z, s_0 \rangle \Rightarrow \langle x := y; y := z, s_0[z \mapsto 5] \rangle$$

a árvore de derivação é

$$\langle z := x, s_0 \rangle \Rightarrow s_0[z \mapsto 5]$$

$$\langle z := x; x := y, s_0 \rangle \Rightarrow \langle x := y, s_0[z \mapsto 5] \rangle$$

$$\langle (z := x; x := y); y := z, s_0 \rangle \Rightarrow \langle x := y; y := z, s_0[z \mapsto 5] \rangle$$

Construa as árvores de derivação para os restantes passos da sequência.

60

Indução no comprimento das sequências

Provar uma propriedade para todas as sequências de derivação.

Caso de base

- Provar que a propriedade se verifica para sequências de derivação de comprimento 0.

Passo indutivo

- Assumir que a propriedade se verifica para todas as sequências de derivação de comprimento até k (é a *hipótese de indução*) e provar que a propriedade também se verifica para as sequências de derivação de comprimento $k+1$.

62

Sequência de Derivação

Exercício: Considere o seguinte programa

$$z:=0; \text{while } y \leq x \text{ do } (z:=z+1; x:=x-y)$$

Determine um estado para o qual a sequência de derivação deste programa é finita e outro para o qual é infinita.

- A execução de um programa S num estado s
 - termina** se e só se existir uma sequência de derivação finita começada em $\langle S, s \rangle$
 - diverge** (ou **entra em ciclo**) se e só se existir uma sequência de derivação infinita começada em $\langle S, s \rangle$
 - termina com sucesso** se e só se $\langle S, s \rangle \Rightarrow^* s'$

Nota: na linguagem **While** não temos configurações que bloqueiem, mas veremos extensões que sim.

61

Propriedades

Lema: Se $\langle S_1, s \rangle \Rightarrow^k s'$ então $\langle S_1; S_2, s \rangle \Rightarrow^k \langle S_2, s' \rangle$

Prova: Por indução em k .

Lema: Se $\langle S_1; S_2, s \rangle \Rightarrow^k s''$ então existe um estado s' e números naturais k_1 e k_2 tais que $\langle S_1, s \rangle \Rightarrow^{k_1} s', \langle S_2, s' \rangle \Rightarrow^{k_2} s''$ e $k = k_1 + k_2$.

Prova: Por indução no comprimento da sequência $\langle S_1; S_2, s \rangle \Rightarrow^k s''$.

63

Determinismo

A semântica operacional estrutural aqui apresentada é *determinista*.

Teorema: Para quaisquer S , s , γ e γ' ,

se $\langle S, s \rangle \Rightarrow \gamma$ e $\langle S, s \rangle \Rightarrow \gamma'$ então $\gamma = \gamma'$.

Prova: Por indução na estrutura da derivação.

64

A função semântica S_{sos}

O significado de um programa pode ser visto como uma função parcial de **State** para **State**.

Definição: $S_{\text{sos}}: \text{Stm} \rightarrow (\text{State} \leftrightarrow \text{State})$

$$S_{\text{sos}}[S]s = \begin{cases} s' & \text{if } \langle S, s \rangle \Rightarrow^* s' \\ \text{undef} & \text{otherwise} \end{cases}$$

A boa definição de S_{sos} é uma consequência do determinismo da relação de transição.

66

Equivalência Semântica

Definição: Dois programas S_1 e S_2 dizem-se *semanticamente equivalentes* se, para todo o estado s ,

- $\langle S_1, s \rangle \Rightarrow^* \gamma$ sse $\langle S_2, s \rangle \Rightarrow^* \gamma$, caso γ seja uma configuração terminal ou bloqueada
- houver uma sequência de derivação começada em $\langle S_1, s \rangle$ divergente sse houver uma começada em $\langle S_2, s \rangle$ também divergente.

Exercício: Mostre que os programas seguintes são semanticamente equivalentes:

- $S;\text{skip}$ and S
- $\text{while } b \text{ do } S \text{ and if } b \text{ then } (S; \text{while } b \text{ do } S) \text{ else skip}$
- $S_1;(S_2;S_3)$ and $(S_1;S_2);S_3$

65

Semântica de transições para expressões

Podemos adoptar uma semântica operacional *small-step* para dar semântica às expressões aritméticas e booleanas. Para isso temos que considerar

$$\text{State} = \text{Var} \rightarrow \text{Num}$$

Exercício: Defina uma semântica de transições para expressões aritméticas.

Exercício: Defina uma semântica de transições para expressões booleanas.

Com poderia fazer uma avaliação “*curto-circuito*” das expressões booleanas (ao estilo do C) ?

67

Tratamento de erros

Considere que queremos estender a linguagem de expressões aritméticas com a operação de divisão.

Como avaliar a divisão por zero ?

Temos diferentes opções:

- Associar à divisão por zero um valor no domínio de interpretação (considerando assim a operação válida).
- Usando semântica operacional, definir o sistema de inferência das transições de forma a que não seja possível derivar transições de expressões que envolvam divisões por zero.
Teremos assim algumas **configurações bloqueadas**.
- Introduzir um valor especial de **erro**, \perp , e estender os domínios de interpretação das expressões com este valor.

$$\mathbf{Z}_\perp = \mathbf{Z} \cup \{\perp\}$$

68

$$\mathbf{T}_\perp = \mathbf{T} \cup \{\perp\}$$

Tratamento de erros

Exercício:

- Defina uma semântica operacional para expressões aritméticas que assinala erro no caso de ocorrências de divisões por zero.
- Faça o mesmo para avaliação de expressões booleanas.
- Experimente as várias opções sugeridas.

69

Execução de programas com ocorrência de erros

Usando semântica operacional, podemos lidar com esta situação de duas formas:

- Definir o sistema de inferência das transições de forma a que comandos avaliados em estados que deem origem a expressões erróneas venham a dar origem a **configurações bloqueadas**.
- Considerar a existência de um **estado de erro**, \perp , e considerar como estados finais possíveis os elementos do conjunto

$$\mathbf{State}_\perp = \mathbf{State} \cup \{\perp\}$$

Os estados onde os comandos podem ser executados são os do conjunto **State**.

70

Execução de programas com ocorrência de erros

Exercício:

- Escreva as regras de uma versão da semântica de transições small-step dos comandos que lide com situações de erro.
- Experimente as várias opções sugeridas.

71