

Strukturelle operationale Semantik

- Grundidee: Spezifiere, wie die Ergebnisse der Einzelschritte der Berechnung zustandekommen.
- Übergangssystem: (Γ, E, \Rightarrow) mit
 - $\Gamma = \{(P, s) \mid P \in \text{WHILE}, s \in \text{State}\} \cup \text{State}$
 - $E = \text{State}$
 - $\Rightarrow \subseteq \{(P, s) \mid P \in \text{WHILE}, s \in \text{State}\} \times \Gamma$
- Zwei typischer Übergänge:
 - Berechnung nach einem Schritt noch nicht zu Ende:

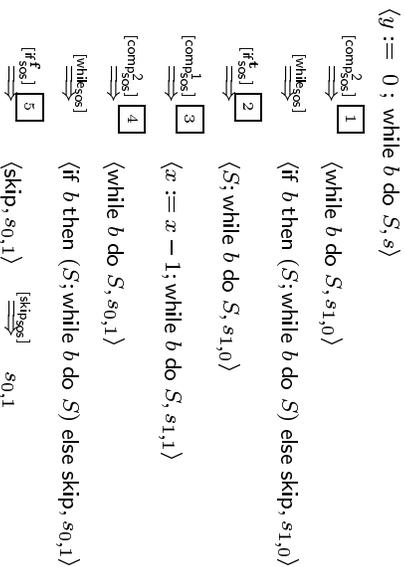
$$\langle P, s \rangle \Rightarrow \langle P', s' \rangle$$
 - Berechnung nach einem Schritt zu Ende:

$$\langle P, s \rangle \Rightarrow s'$$

21

Beispiel: strukturelle operationale Semantik

prog. $P: y := 0; \text{while } \underbrace{\neg(x=0)}_b \text{ do } \underbrace{(y := y+x; x := x-1)}_{S_1 \quad S_2}$
 leg.: s mit $sv = 1$. Ges.: s' mit $\langle P, s \rangle \Rightarrow^* s'$. Not.: s_m, n wie zuvor.



23

Strukturelle operationale Semantik: Inferenzregelsystem

- $$\begin{array}{l} [\text{ass}_{\text{SOS}}] \quad \langle x := a, s \rangle \Rightarrow s[x \mapsto \mathcal{A}[a]]s \\ [\text{skip}_{\text{SOS}}] \quad \langle \text{skip}, s \rangle \Rightarrow s \\ [\text{comp}_{\text{SOS}}^1] \quad \frac{\langle P_1, s \rangle \Rightarrow \langle P'_1, s' \rangle}{\langle P_1; P_2, s \rangle \Rightarrow \langle P'_1; P_2, s' \rangle} \\ [\text{comp}_{\text{SOS}}^2] \quad \frac{\langle P_1, s \rangle \Rightarrow s'}{\langle P_1; P_2, s \rangle \Rightarrow \langle P_2, s' \rangle} \\ [\text{if}_{\text{SOS}}^t] \quad \langle \text{if } b \text{ then } P_1 \text{ else } P_2, s \rangle \Rightarrow \langle P_1, s \rangle \text{ falls } \mathcal{B}[b]s = t \\ [\text{if}_{\text{SOS}}^f] \quad \langle \text{if } b \text{ then } P_1 \text{ else } P_2, s \rangle \Rightarrow \langle P_2, s \rangle \text{ falls } \mathcal{B}[b]s = f \\ [\text{while}_{\text{SOS}}] \quad \langle \text{while } b \text{ do } P, s \rangle \Rightarrow \langle \text{if } b \text{ then } (P; \text{while } b \text{ do } P) \text{ else skip}, s \rangle \end{array}$$

22

Beispiel: Rechtfertigung für Einzelschritte

- $$\begin{array}{l} \boxed{1} \quad \frac{\langle y := 0, s \rangle \xrightarrow{[\text{ass}_{\text{SOS}}]} s_{1,0}}{\langle y := 0; \text{while } b \text{ do } S, s \rangle \Rightarrow \langle \text{while } b \text{ do } S, s_{1,0} \rangle} [\text{comp}_{\text{SOS}}^2] \\ \boxed{2} \quad \mathcal{B}[\neg(x=0)]_{s_{1,0}} = t \\ \quad \frac{\langle y := y+x, s_{1,0} \rangle \xrightarrow{[\text{ass}_{\text{SOS}}]} s_{1,1}}{\langle y := y+x; x := x-1, s_{1,0} \rangle \Rightarrow \langle x := x-1, s_{1,1} \rangle} [\text{comp}_{\text{SOS}}^2] \\ \boxed{3} \quad \frac{\langle S; \text{while } b \text{ do } S, s_{1,0} \rangle \Rightarrow \langle x := x-1; \text{while } b \text{ do } S, s_{1,1} \rangle}{\langle x := x-1, s_{1,1} \rangle \xrightarrow{[\text{ass}_{\text{SOS}}]} s_{0,1}} [\text{comp}_{\text{SOS}}^1] \\ \boxed{4} \quad \frac{\langle x := x-1; \text{while } b \text{ do } S, s_{1,1} \rangle \Rightarrow \langle \text{while } b \text{ do } S, s_{0,1} \rangle}{\langle x := x-1, s_{1,1} \rangle \xrightarrow{[\text{ass}_{\text{SOS}}]} s_{0,1}} [\text{comp}_{\text{SOS}}^2] \\ \boxed{5} \quad \mathcal{B}[\neg(x=0)]_{s_{0,1}} = f \end{array}$$

24

Ableitungen in struktureller operationaler Semantik

- Ableitungsbäume für Einzelschritte der Form
 - $\langle P, s \rangle \Rightarrow \langle P', s' \rangle$ oder
 - $\langle P, s \rangle \Rightarrow s'$
- Ableitungen (Ableitungsketten, -sequenzen) der Form
 - $\langle P, s \rangle \Rightarrow \langle P_1, s_1 \rangle \Rightarrow \langle P_2, s_2 \rangle \Rightarrow \dots$ oder
 - $\langle P, s \rangle \Rightarrow \langle P_1, s_1 \rangle \Rightarrow \dots \Rightarrow \langle P_{n-1}, s_{n-1} \rangle \Rightarrow s_n$
- Ableitungen können endlich oder unendlich sein
- Konfigurationen γ :
 - γ *reduzibel* falls ein γ' existiert mit $\gamma \Rightarrow \gamma'$
 - γ *irreduzibel* sonst ($\gamma \not\Rightarrow$)
 Terminologie: γ *hängt (bleibt stecken)*

25

Semantische Äquivalenz (von Programmen)

- Natürliche Semantik (NS):

P_1 und P_2 sind *semantisch äquivalent*, falls für alle Zustände s, s' gilt:

$$\langle P_1, s \rangle \rightarrow s' \quad \text{gdw.} \quad \langle P_2, s \rangle \rightarrow s'$$
- Strukturelle operationale Semantik (SoS):
 - P_1 und P_2 sind *semantisch äquivalent*, falls für alle s gilt:

Für alle Konfigurationen γ , die hängen oder Endkonfiguration sind:

$$\langle P_1, s \rangle \Rightarrow^* \gamma \quad \text{gdw.} \quad \langle P_2, s \rangle \Rightarrow^* \gamma$$
 - Es gibt eine unendliche Ableitung ausgehend von $\langle P_1, s \rangle$

gdw. es eine solche ausgehend von $\langle P_2, s \rangle$ gibt

27

Nichtterminierung von Programmen

Wie wird Nichtterminierung semantisch repräsentiert?

- Natürliche Semantik (NS):

Nicht alle Programme P , ausgehend vom Zustand s , besitzen einen Ableitungsbaum: $\langle P, s \rangle \not\Rightarrow$

$$\langle \text{while true do skip}, s \rangle \not\Rightarrow \quad (s \text{ beliebig})$$
- Strukturelle operationale Semantik (SoS):

Nicht alle Programme P , ausgehend vom Zustand s , besitzen eine endliche Ableitungskette:

$$\begin{aligned} &\langle \text{while true do skip}, s \rangle \\ &\quad \Rightarrow^* \langle \text{while true do skip}, s \rangle \\ &\quad \Rightarrow \dots \end{aligned}$$

26

Semantische Äquivalenz: Beispiel

Lemma: $P_1; (P_2; P_3)$ und $(P_1; P_2); P_3$ sind semantisch äquivalent.

Beweis (idee) in NS:
Gegeben

$$\begin{array}{c} \langle P_2, s_2 \rangle \rightarrow s_3 \quad \langle P_3, s_3 \rangle \rightarrow s_4 \\ \hline \langle P_2; P_3, s_2 \rangle \rightarrow s_4 \\ \hline \langle P_1; (P_2; P_3), s_1 \rangle \rightarrow s_4 \end{array}$$

konstruiere

$$\begin{array}{c} \langle P_1, s_1 \rangle \rightarrow s_2 \quad \langle P_2, s_2 \rangle \rightarrow s_3 \\ \hline \langle P_1; P_2, s_1 \rangle \rightarrow s_3 \\ \hline \langle P_1; P_2; P_3, s_1 \rangle \rightarrow s_4 \end{array}$$

und umgekehrt.

28

Definitions- und Beweisprinzipien

Sätze für Semantik-Definitionen:

- kompositionale Definitionen
 - natürliche Semantik
 - strukturelle operationale Semantik
- entsprechende Beweisprinzipien:

- **strukturelle Induktion**
- **(strukturelle) Induktion über Ableitungsbäume:**
 - IA: Beweis für elementare Ableitungsbäume (Axiome)
 - IS: Beweis für zusammengesetzte Ableitungsbäume (Regeln)
- **(natürliche) Induktion über Ableitungslänge:**
 - IA: Beweis für alle Ableitungen der Länge 0
 - IS: Beweis von " $k \Rightarrow k + 1$ "

29

Semantik arithmetischer Ausdrücke (zur Erinnerung)

- $\mathcal{A} : \text{Aexp} \rightarrow (\text{State} \rightarrow \mathbb{Z})$

$$\begin{aligned}\mathcal{A}[[n]]s &= \mathcal{N}[[n]] \\ \mathcal{A}[[x]]s &= s[x] \\ \mathcal{A}[[a_1 + a_2]]s &= \mathcal{A}[[a_1]]s + \mathcal{A}[[a_2]]s \\ \mathcal{A}[[a_1 * a_2]]s &= \mathcal{A}[[a_1]]s * \mathcal{A}[[a_2]]s \\ \mathcal{A}[[a_1 - a_2]]s &= \mathcal{A}[[a_1]]s - \mathcal{A}[[a_2]]s\end{aligned}$$

31

Strukturelle Induktion: Beispiel

Intuitiv: Der Wert eines arithmetischen Ausdrucks hängt nur von den Werten derjenigen Variablen ab, die in ihm auftreten.

Formalisierung (freie Variablen):

$$\begin{aligned}\text{FV}(n) &= \emptyset \\ \text{FV}(x) &= \{x\} \\ \text{FV}(a_1 + a_2) &= \text{FV}(a_1) \cup \text{FV}(a_2) \\ \text{FV}(a_1 * a_2) &= \text{FV}(a_1) \cup \text{FV}(a_2) \\ \text{FV}(a_1 - a_2) &= \text{FV}(a_1) \cup \text{FV}(a_2)\end{aligned}$$

Lemma: Seien s, s' Zustände mit $sx = s'x$ für alle $x \in \text{FV}(a)$.

Dann gilt:

$$\mathcal{A}[[a]]s = \mathcal{A}[[a]]s'.$$

Beweis: Strukturelle Induktion über arithmetische Ausdrücke.

30

Strukturelle Induktion über Ableitungsbäume: Beispiel

Satz: Die natürliche Semantik der Sprache **WHILE** ist deterministisch, d.h.: Für alle Programme P in **WHILE** und alle Zustände s, s' und s'' gilt:

Falls $\langle P, s \rangle \rightarrow s'$ und $\langle P, s \rangle \rightarrow s''$, dann folgt $s' = s''$.

Beweisstruktur:

- Annahme: $\langle P, s \rangle \rightarrow s'$.
- Zeige, daß aus $\langle P, s \rangle \rightarrow s''$ folgt: $s' = s''$.
- Verwende dafür Induktion über den Ableitungsbau von $\langle P, s \rangle \rightarrow s'$.

32

Induktion über Ableitungslänge: Beispiel

Intuitiv: Ableitungen von zusammengesetzten Programmen $P_1; P_2, s$ in SoS zu einem Endzustand lassen sich in zwei Teile zerlegen. Genauer:

Lemma: Falls $\langle P_1; P_2, s \rangle \Rightarrow^k s''$ gilt, dann gibt es einen Zustand s' und natürliche Zahlen k_1, k_2 mit

- $\langle P_1, s \rangle \Rightarrow^{k_1} s'$,

- $\langle P_2, s' \rangle \Rightarrow^{k_2} s''$ und

- $k = k_1 + k_2$.

Beweisstruktur:

Induktion über die Ableitungslänge k in $\langle P_1; P_2, s \rangle \Rightarrow^k s''$.

33

Hilfslemmata, weitere Eigenschaften

Übung (Teilprogrammabarbeitung in SoS):

Aus $\langle P_1, s \rangle \Rightarrow^k s'$ folgt $\langle P_1; P_2, s \rangle \Rightarrow^k \langle P_2, s' \rangle$

Lemma (Zerlegung, siehe oben): Falls $\langle P_1; P_2, s \rangle \Rightarrow^k s''$ gilt, dann gibt es einen Zustand s' und natürliche Zahlen k_1, k_2 mit

- $\langle P_1, s \rangle \Rightarrow^{k_1} s'$,

- $\langle P_2, s' \rangle \Rightarrow^{k_2} s''$ und

- $k = k_1 + k_2$.

Übung (einfache Äquivalenzen):

Die Programme

while b do P

und

if b then $(P; \text{while } b \text{ do } P)$ else skip

sind semantisch äquivalent.

35

Semantische Funktionen, Vergleich

Natürliche Semantik:

$$S_{ns}[P]s = \begin{cases} s', & \text{falls } \langle P, s \rangle \rightarrow s' \\ \text{undefiniert,} & \text{sonst} \end{cases}$$

Strukturelle operationale Semantik:

$$S_{sos}[P]s = \begin{cases} s', & \text{falls } \langle P, s \rangle \Rightarrow^* s' \\ \text{undefiniert,} & \text{sonst} \end{cases}$$

Frage: Gilt $S_{ns} = S_{sos}$?

Lemma A: Für alle P, s, s' gilt: $\langle P, s \rangle \rightarrow s'$ impliziert $\langle P, s \rangle \Rightarrow^* s'$.
Beweisstruktur:

Induktion über die Struktur des Ableitungsbaumes von $\langle P, s \rangle \rightarrow s'$.

Lemma B: Für alle P, s, s' gilt: $\langle P, s \rangle \Rightarrow^* s'$ impliziert $\langle P, s \rangle \rightarrow s'$.
Beweisstruktur:

Induktion über die Länge der Ableitung $\langle P, s \rangle \Rightarrow^* s'$.

34

Spracherweiterungen von WHILE

Neue Konstrukte:

$$P ::= x := a \mid \text{skip} \mid S_1; S_2 \mid \text{if } b \text{ then } S_1 \text{ else } S_2 \mid \text{while } b \text{ do } S \\ \mid \text{abort} \\ \mid P_1 \text{ or } P_2 \\ \mid P_1 \text{ par } P_2$$

- Wie wird die entsprechende Semantik (in NS und SoS) definiert?

- Seide NS und SoS “gleich mächtig” / “gleich geeignet”?

36

Spracherweiterung: Abbruchoperation abort

- Konfigurationen:

$$\{\langle P, s \rangle \mid P \in \mathbf{WHILE}^{\text{abort}}, s \in \text{State}\} \cup \text{State}$$

- Übergangsrelation für NS: wie zuvor
- Übergangsrelation für SoS: wie zuvor
- Alternative(n)?

37

Spracherweiterung: Nichtdeterminismus

- Konfigurationen:

$$\{\langle P, s \rangle \mid P \in \mathbf{WHILE}^{\text{or}}, s \in \text{State}\} \cup \text{State}$$

- Übergangsrelation für NS:

$$\frac{\langle P_1, s \rangle \rightarrow s'}{\langle P_1 \text{ or } P_2, s \rangle \rightarrow s'}$$

$$\frac{\langle P_2, s \rangle \rightarrow s'}{\langle P_1 \text{ or } P_2, s \rangle \rightarrow s'}$$

$$\frac{\langle P_1, s \rangle \rightarrow s' \quad \langle P_2, s \rangle \rightarrow s'}{\langle P_1 \text{ or } P_2, s \rangle \rightarrow s'}$$

- Übergangsrelation für SoS:

$$\langle P_1 \text{ or } P_2, s \rangle \Rightarrow \langle P_1, s \rangle$$

$$\langle P_1 \text{ or } P_2, s \rangle \Rightarrow \langle P_2, s \rangle$$

38

Spracherweiterungen: Parallelismus (1)

- Konfigurationen:

$$\{\langle P, s \rangle \mid P \in \mathbf{WHILE}^{\text{par}}, s \in \text{State}\} \cup \text{State}$$

- Übergangsrelation für SoS:

$$\frac{\langle P_1, s \rangle \Rightarrow \langle P'_1, s' \rangle}{\langle P_1 \text{ par } P_2, s \rangle \Rightarrow \langle P'_1 \text{ par } P_2, s' \rangle}$$

$$\frac{\langle P_1, s \rangle \Rightarrow s'}{\langle P_1 \text{ par } P_2, s \rangle \Rightarrow \langle P_2, s' \rangle}$$

$$\frac{\langle P_2, s \rangle \Rightarrow s'}{\langle P_1 \text{ par } P_2, s \rangle \Rightarrow \langle P_1, s' \rangle}$$

$$\frac{\langle P_2, s \rangle \Rightarrow \langle P'_2, s' \rangle}{\langle P_1 \text{ par } P_2, s \rangle \Rightarrow \langle P_1 \text{ par } P'_2, s' \rangle}$$

$$\frac{\langle P_1 \text{ par } P_2, s \rangle \Rightarrow \langle P_1, s' \rangle}{\langle P_1 \text{ par } P_2, s \rangle \Rightarrow \langle P_1, s' \rangle}$$

$$\frac{\langle P_1 \text{ par } P_2, s \rangle \Rightarrow s'}{\langle P_1 \text{ par } P_2, s \rangle \Rightarrow \langle P_1, s' \rangle}$$

$$\frac{\langle P_1 \text{ par } P_2, s \rangle \Rightarrow \langle P_1, s' \rangle}{\langle P_1 \text{ par } P_2, s \rangle \Rightarrow \langle P_1, s' \rangle}$$

39

Spracherweiterungen: Parallelismus (2)

- Übergangsrelation für NS:

$$\frac{\langle P_1, s \rangle \rightarrow s' \quad \langle P_2, s \rangle \rightarrow s''}{\langle P_1 \text{ par } P_2, s \rangle \rightarrow s''}$$

$$\frac{\langle P_1, s \rangle \rightarrow s' \quad \langle P_2, s \rangle \rightarrow s''}{\langle P_1 \text{ par } P_2, s \rangle \rightarrow s''}$$

$$\frac{\langle P_2, s \rangle \rightarrow s' \quad \langle P_1, s \rangle \rightarrow s''}{\langle P_1 \text{ par } P_2, s \rangle \rightarrow s''}$$

$$\frac{\langle P_1, s \rangle \rightarrow s' \quad \langle P_2, s \rangle \rightarrow s''}{\langle P_1 \text{ par } P_2, s \rangle \rightarrow s''}$$

40

Öbpracherweiterungen: Zusammenfassung

Natürliche Semantik:

- keine Unterscheidung zwischen Nichtterminierung und Abbruch (mit abort)
 - Nichtdeterminismus kann Nichtterminierung unterdrücken
 - keine verzahnten Berechnungen möglich (*no interleaving*)
- ### Strukturelle operationale Semantik:
- Unterscheidung zwischen Nichtterminierung und und Abbruch (mit abort)
 - Nichtdeterminismus kann Nichtterminierung nicht verhindern
 - Verzahnen (*interleaving*) von parallelen Berechnungen (beliebig fein) möglich (*Sos ist Einzelschrittsemantik(!)*)

41

Operationale Semantik: Ausblick

- Die Semantik von abort kann auch alternativ (adäquater) definiert werden, mittels neuem *Abbruchzustand* STOP.
- Die Semantik arithmetischer / boolescher Ausdrücke kann ebenfalls leicht in NS und SoS spezifiziert werden (Übung!).
- Semantikspezifikationen anderer programmiersprachlicher Konstrukte (andere Schleifen, Deklarationen, (rekursive) Prozeduren, ...) funktionieren **im Prinzip analog**, können aber **im Detail bedeutend komplexer** werden.
- Eine relativ abstrakte, inferenzregelbasierte Spezifikation (wie bei NS und SoS) ermöglicht ein konzeptionelles Verständnis und Vergleiche (z.B. zwischen verschiedenen Varianten) auf relativ hohem Niveau.
- Die Semantik von Sprachkonstrukten ist nicht *a priori* festgelegt, sondern muß möglichst adäquat definiert werden!

42