2.2.2 Big-Step-Semantik

Ziel: Definiere eine gesamtheitliche Semantik für W--, die ganze Ausführungen von einer Initialkonfiguration zu einem Endzustand zusammenfasst.

Definition:

Die Big-Step-Transitionsrelation

$$\Downarrow \subseteq \underbrace{(Prog \times State)}_{\text{Konfiguration}} \times \underbrace{State}_{\text{Zustand}}$$

ist die kleinste Relation, die folgende Regeln genügt:

$$(\text{BSKIP})\ \frac{}{(skip, \sigma) \Downarrow \sigma} \qquad \frac{}{(x = a, \sigma) \Downarrow \sigma[x \mapsto S[\![a]\!]\sigma]}\ (\text{BASSIGN})$$

$$(\text{BSEQ})\ \frac{(c_1, \sigma) \Downarrow \sigma' \qquad (c_2, \sigma') \Downarrow \sigma''}{(c_1; c_2, \sigma) \Downarrow \sigma''}$$

$$(\text{BIF\_TRUE})\ \frac{(c_1, \sigma) \Downarrow \sigma'}{(\underline{if}\ b\ \underline{then}\ c_1\ \underline{else}\ c_2\ \underline{fi}, \sigma) \Downarrow \sigma'}, \text{ falls } S[\![b]\!]\sigma = 1$$

$$(\text{BIF\_FALSE})\ \frac{(c_2, \sigma) \Downarrow \sigma'}{(\underline{if}\ b\ \underline{then}\ c_1\ \underline{else}\ c_2\ \underline{fi}, \sigma) \Downarrow \sigma'}, \text{ falls } S[\![b]\!]\sigma = 0$$

$$(\text{BWHILE\_FALSE})\ \frac{}{(\underline{while}\ b\ \underline{do}\ c\ \underline{od}, \sigma) \Downarrow \sigma}, \text{ falls } S[\![b]\!]\sigma = 0$$

$$(\text{BWHILE\_TRUE})\ \frac{(c, \sigma) \Downarrow \sigma' \quad (\underline{while}\ b\ \underline{do}\ c\ \underline{od}, \sigma') \Downarrow \sigma''}{(\underline{while}\ b\ \underline{do}\ c\ \underline{od}, \sigma) \Downarrow \sigma''}, \text{ falls } S[\![b]\!]\sigma = 1$$

$$(\text{BASSUME})\ \frac{}{(\underline{assume}\ b, \sigma) \Downarrow \sigma}, \text{ falls } S[\![b]\!]\sigma = 1$$

Beispiel:

Betrachte wieder das Programm

$$c \equiv x := x+1 \,;\, loop$$

mit $loop \equiv \underline{\text{while}}\ x>0 \wedge y<1\ \underline{\text{do}}\ y:=y+1\ \underline{\text{od}}$

Dann erhalten wir folgenden Beweisbaum:

$$
\text{(BSEQ)}\ \dfrac{
\text{(BASSIGN)}\ \dfrac{}{(x:=x+1,(0,0)) \Downarrow (1,0)}
\qquad
(*)\ \dfrac{
\text{(BASSIGN)}\ \dfrac{}{(y:=y+1,(1,0)) \Downarrow (1,1)}
\qquad
(**)\ \dfrac{}{(loop,(1,1)) \Downarrow (1,1)}
}{(loop,(1,0)) \Downarrow (1,1)}
}{(c,(0,0)) \Downarrow (1,1)}
$$

(*) (BWHILE-TRUE), denn $\mathcal{S}[\![x>0 \wedge y<1]\!](1,0) = 1$.

(**) (BWHILE-FALSE), denn $\mathcal{S}[\![x>0 \wedge y<1]\!](1,1) = 0$.

Beobachtung:

- In der Big-Step-Semantik kann Divergenz / Nicht-Terminierung nicht erkannt werden.
  Es gibt lediglich keinen finalen Zustand, da der Beweisbaum nicht abgeschlossen werden kann.
- Im Gegensatz dazu ist in der Small-Step-Semantik Divergenz erkennbar, da die Zwischenkonfigurationen sichtbar sind.
  Das Programm divergiert genau dann, wenn es ein unendliches Transitionssystem oder ein endliches Transitionssystem mit einem Kreis hat.

Da $\to^*$ und $\Downarrow$ beide über ganze Berechnungen sprechen, ist folgender Zusammenhang wenig überraschend:

<u>Satz (Zusammenhang Small-Step und Big-Step-Semantik)</u>:
Für alle c, σ, σ' gilt: $(c,\sigma) \to^* \sigma'$ gdw. $(c,\sigma) \Downarrow \sigma'$.

<u>Beweis (Idee)</u>:
„$\Rightarrow$" Induktion nach der Länge n der Berechnung.

<u>Zeige</u>:

$$\forall c \in W\text{--}\ \forall \sigma, \sigma' \in State.\ (c,\sigma) \to^n \sigma' \Rightarrow (c,\sigma) \Downarrow \sigma'.$$

<u>IA</u>: Betrachte Berechnungen der Länge 1.
Um bei einem Zustand σ' zu landen,
kommen nur die Axiome

(SKIP), (ASSIGN), (WHILE-FALSE) und (ASSUME)

in Frage.
Diese haben Entsprechungen in der Big-Step-Semantik.

<u>IS</u>: Betrachte eine Ableitung der Länge $n+1$:

$$(c_0, \sigma_0) \to (c, \sigma) \to^n \sigma'.$$

Nach <u>Induktionsvoraussetzung</u> gilt $(c,\sigma) \Downarrow \sigma'$.
Führe nun eine Fallunterscheidung
über die möglichen Transitionen $(c_0, \sigma_0) \to (c, \sigma)$ durch.

„$\Leftarrow$" Führe eine Noethersche Induktion nach der Höhe des Beweisbaums.

<u>Zeige</u>: Falls $(c,\sigma) \Downarrow \sigma'$ mit einem Beweisbaum der Höhe <u>höchstens n</u>
nachgewiesen werden kann,
gibt es eine Berechnung $(c,\sigma) \to^* \sigma'$. $\square$

Vergleich von Small-Step und Big-Step-Semantik

Small-Step:

- Komplexe Programmierkonstrukte sind einfach zu modellieren:
  Nebenläufigkeit, Stack, Heap, Divergenz/Nicht-Terminierung,
  Laufzeitfehler, Exceptions.
- Aufwendig Eigenschaften von Programmen nachzuweisen.

Big-Step:

- Typischerweise einfacher, Eigenschaften nachzuweisen, da wenige Beweisregeln.
- Zwischenzustände werden nicht dargestellt,
  dadurch sehen Programme ohne Endzustände gleich aus:

  Schleifen = Errors = Deadlocks.

  Einige Programmeigenschaften können daher
  in der Big-Step-Semantik nicht nachgewiesen werden.

Bemerkung:

- SOS-Semantiken heißen auch natürliche Semantiken,
  da die Beweissysteme den Stil der natürlichen Deduktion haben.
- Die Semantikklammern $S[\![1+1]\!]\sigma$ haben ihren Ursprung darin,
  dass man den inneren Term in Anführungsstriche setzt:

  $S[\![1+1]\!]\sigma$ steht für $S(\text{"}1+1\text{"})\sigma$.

  Die Anführungsstriche wiederum sagen,
  dass das Stück Syntax 1+1 gemapped
  nicht der arithmetische Ausdruck 1+1 evaluiert wird.